K 7

```
RRRRRRRRRRRR     UUU         UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU         UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU         UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRR         RRR  UUU         UUU  NNN         NNN  000         000  FFF              FFF
RRR         RRR  UUU         UUU  NNN         NNN  000         000  FFF              FFF
RRR         RRR  UUU         UUU  NNN         NNN  000         000  FFF              FFF
RRR         RRR  UUU         UUU  NNNNNN      NNN  000         000  FFF              FFF
RRR         RRR  UUU         UUU  NNNNNN      NNN  000         000  FFF              FFF
RRR         RRR  UUU         UUU  NNNNNN      NNN  000         000  FFF              FFF
RRRRRRRRRRRR     UUU         UUU  NNN   NNN   NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU         UUU  NNN   NNN   NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU         UUU  NNN   NNN   NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRR   RRR        UUU         UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR   RRR        UUU         UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR   RRR        UUU         UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR      RRR     UUU         UUU  NNN         NNN  000         000  FFF              FFF
RRR      RRR     UUU         UUU  NNN         NNN  000         000  FFF              FFF
RRR      RRR     UUU         UUU  NNN         NNN  000         000  FFF              FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
```

```
_$2

Sym
---
NDX
NDX
NUM
NUM
OPE
OUT
PAC
PAC
PAC
PAC
PAC
PAC
PAC
PAD
PAG
PAG
PAG
PAG
PAG
PAG
PAG
PER
PUT
RCO
RIN

RLI
RNO
RNO
RTY
SAV
STR
STR
STR
STR
STR
STR

STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
```

```
**FILE**ID**NDXVMS

NN      NN  DDDDDDDD    XX      XX  VV      VV  MM      MM    SSSSSSSS
NN      NN  DDDDDDDD    XX      XX  VV      VV  MM      MM    SSSSSSSS
NN      NN  DD      DD  XX      XX  VV      VV  MMMM  MMMM  SS
NN      NN  DD      DD  XX      XX  VV      VV  MMMM  MMMM  SS
NNNN    NN  DD      DD    XX  XX    VV      VV  MM  MM  MM  SS
NNNN    NN  DD      DD    XX  XX    VV      VV  MM  MM  MM  SS
NN  NN  NN  DD      DD      XX      VV      VV  MM      MM    SSSSSS
NN  NN  NN  DD      DD      XX      VV      VV  MM      MM    SSSSSS
NN    NNNN  DD      DD    XX  XX    VV      VV  MM      MM          SS
NN    NNNN  DD      DD    XX  XX    VV      VV  MM      MM          SS
NN      NN  DD      DD  XX      XX    VV  VV    MM      MM          SS    ....
NN      NN  DD      DD  XX      XX    VV  VV    MM      MM          SS    ....
NN      NN  DDDDDDDD    XX      XX      VV      MM      MM  SSSSSSSS      ....
NN      NN  DDDDDDDD    XX      XX      VV      MM      MM  SSSSSSSS      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

ND
VO

```
0 %TITLE 'NDXVMS -- DSRINDEX/INDEX Command line interface'
0 MODULE NDXVMS    (IDENT = 'V04-000', LANGUAGE (BLISS32),
0                  ADDRESSING_MODE (EXTERNAL    = LONG_RELATIVE,
0                                   NONEXTERNAL = LONG_RELATIVE)
0                  ) =
0
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:
1 !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
1 !
1 ! ABSTRACT:
1 !
1 !       This module is the INDEX command line interface module.
1 !
1 !       Much of the code to parse and validate qualifier
1 !       values may be removed when the VMS CLI interface routines
1 !       implement value validation.
1 !
1 ! ENVIRONMENT:   VAX/VMS User Mode
1 !
1 ! AUTHOR:        JPK
1 !
1 ! CREATION DATE: February-1982
1 !
1 ! MODIFIED BY:
1 !
1 !       012     JPK00023        20-May-1983
1 !               Modified INDEX, NDXT20 and NDXVMS to check status of
1 !               $XPO_PARSE_SPEC to avoid error messages from XPORT.
1 !
```

```
1 !        011     JPK00022        30-Mar-1983
1 !                Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
1 !                to generate TEX output. Added module NDXTEX.
1 !
1 !        010     JPK00019        14-MAR-1983
1 !                Modified NDXVMS to conditionalize /PAGE_NUMBERS=[NO]MERGE
1 !                and /PAGE_NUMBERS=STANDARD for DSRPLUS only.
1 !
1 !        009     JPK00016        23-Feb-1983
1 !                Modified NDXVMS to change the default number of lines per page
1 !                when /TELLTALE is specified but /LINES is not.
1 !
1 !        008     JPK00015        04-Feb-1983
1 !                Cleaned up module names, modified revision history to
1 !                conform with established standards. Updated copyright dates.
1 !
1 !        007     JPK00013        31-Jan-1983
1 !                Changed default subindex level value from 6 to 99 in NDXVMS
1 !                and NDXCLIDMP. This value is the subindexing level.
1 !                It is NOT A HEADER LEVEL.
1 !
1 !        006     JPK00012        24-Jan-1983
1 !                Modified NDXVMSMSG.MSG to define error messages for both
1 !                DSRINDEX and INDEX.
1 !                Added require of NDXVMSREQ.R32 to NDXOUT, NDXFMT, NDXDAT,
1 !                INDEX, NDXMSG, NDXXTN, NDXTMS, NDXVMS and NDXPAG for BLISS32.
1 !                Since this file defines the error message literals,
1 !                the EXTERNAL REFERENCEs for the error message literals
1 !                have been removed.
1 !
1 !        005     JPK00011        24-Jan-1983
1 !                Changed CMDBLK [NDX$G_LEVEL] to CMDBLK [NDX$H_LEVEL]
1 !                Changed CMDBLK [NDX$H_FORMAT] to CMDBLK [NDX$H_LAYOUT]
1 !                Changed CMDBLK [NDX$V_TMS11] and CMDBLK [NDX$V_TEX] to CMDBLK [NDX$H_FORMAT]
1 !                Changed comparisons of (.CHRSIZ EQLA CHRSZA) to
1 !                (.CMDBLK [NDX$H_FORMAT] EQL TMS11_A).
1 !                Definitions were changed in NDXCLI and references to the
1 !                effected fields were changed in NDXPAG, NDXFMT, INDEX, NDXVMS
1 !                and NDXCLIDMP.
1 !
1 !        004     RER00002        20-Jan-1983
1 !                Modified VMS command line interface module NDXVMS:
1 !                  - changed /FORMAT qualifier to /LAYOUT.
1 !                  - changed use of /RESERVE and /REQUIRE for DSRPLUS.
1 !                  - added code for new DSRPLUS qualifiers /FORMAT and
1 !                     /TELLTALE_HEADINGS.
1 !                Added fields to NDXCLI for new qualifiers: NDX$V_TELLTALE
1 !                and NDX$V_TEX.
1 !                Conditionalized output of NDX$V_PAGE_MERGE in NDXCLIDMP to
1 !                account for different DSR and DSRPLUS default values.
1 !
1 !        003     RER00001        17-Dec-1982
1 !                Modified VMS command line interface module NDXVMS:
1 !                  - Added code to treat keyword NORUNNING in same way as
1 !                     keyword STANDARD.
1 !                  - Added code for new DSR qualifiers /RESERVE and /REQUIRE.
1 !                  - Changed header level default value from 99 to 6.
```

```
;  115          0115  1 !                   - Conditionalized code to compile for DSRPLUS if BLISS
;  116          0116  1 !                     /VARIANT = 8192 is used; otherwise, to compile for DSR.
;  117          0117  1 !                   - Deleted foreign-command code; INDEX is now called
;  118          0118  1 !                     as a subcommand of DSR.
;  119          0119  1 !
;  120          0120  1 !           002     JPK00001        13-Aug-1982
;  121          0121  1 !                   Removed reference to CLI$END_PARSE in NDXVMS. It is no longer
;  122          0122  1 !                   supported by VMS.
;  123          0123  1 !
;  124          0124  1 !--
;  125          0125  1
;  126          0126  1 !
;  127          0127  1 ! INCLUDE FILES:
;  128          0128  1 !
;  129          0129  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';          ! System macro library
;  130          0130  1
;  131          0131  1 LIBRARY 'SYS$LIBRARY:TPAMAC.L32';           ! TPARSE macros
;  132          0132  1
;  133          0133  1 LIBRARY 'SYS$LIBRARY:XPORT';                ! Transportable BLISS library
;  134          0134  1
;  135          0135  1 SWITCHES LIST (REQUIRE);                    ! Print require files
;  136          0136  1
;  137          0137  1 REQUIRE 'REQ:NDXCLI';                       ! Command line information block
```

```
!                               IDENT = 0V04-00004
!
!****************************************************************************
!*                                                                          *
!*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
!*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
!*   ALL RIGHTS RESERVED.                                                   *
!*                                                                          *
!*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
!*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
!*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
!*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
!*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
!*   TRANSFERRED.                                                           *
!*                                                                          *
!*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
!*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
!*   CORPORATION.                                                           *
!*                                                                          *
!*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
!*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!
!++
! FACILITY:
!   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
!
! ABSTRACT:       INDEX command line definitions
!
! ENVIRONMENT:    Transportable
!
! AUTHOR:         JPK
!
! CREATION DATE: January 1982
!
! MODIFIED BY:
!
!       004     JPK00015        04-Feb-1983
!               Cleaned up module names, modified revision history to
!               conform with established standards. Updated copyright dates.
!
!       003     JPK00011        24-Jan-1983
!               Changed CMDBLK [NDX$G_LEVEL] to CMDBLK [NDX$H_LEVEL]
!               Changed CMDBLK [NDX$H_FORMAT] to CMDBLK [NDX$H_LAYOUT]
!               Changed CMDBLK [NDX$V_TMS11] and CMDBLK [NDX$V_TEX] to CMDBLK [NDX$H_FORMAT]
!               Changed comparisons of (.CHRSIZ EQLA CHRSZA) to
!               (.CMDBLK [NDX$H_FORMAT] EQL TMS11_A).
!               Definitions were changed in NDXCLI and references to the
!               effected fields were changed in NDXPAG, NDXFMT, INDEX, NDXVMS
!               and NDXCLIDMP.
!
!       002     RER00002        20-Jan-1983
!               Modified VMS command line interface module NDXVMS:
!                 - changed /FORMAT qualifier to /LAYOUT.
```

```
;  R0195  1      !                  - changed use of /RESERVE and /REQUIRE for DSRPLUS.
;  R0196  1      !                  - added code for new DSRPLUS qualifiers /FORMAT and
;  R0197  1      !                    /TELLTALE_HEADINGS.
;  R0198  1      !                Added fields to NDXCLI for new qualifiers: NDX$V_TELLTALE
;  R0199  1      !                and NDX$V_TEX.
;  R0200  1      !                Conditionalized output of NDX$V_PAGE_MERGE in NDXCLIDMP to
;  R0201  1      !                account for different DSR and DSRPLUS default values.
;  R0202  1      !
;  R0203  1      !--
;  R0204  1
```

```
!
!   NDXCMD_FIELDS
!
$FIELD ndxcmd_fields =
    SET

    NDX$V_OPTIONS        = [$INTEGER],           ! Command option indicators:

        $OVERLAY (NDX$V_OPTIONS)

        NDX$V_INPUT_CONCAT       = [$BIT],        ! Input file concatenated to previous
        NDX$V_OUTPUT             = [$BIT],        ! Generate output file
        NDX$V_REQUIRE            = [$BIT],        ! Require file specified
        NDX$V_PAGES              = [$BIT],        ! Include page references in index
        NDX$V_OVERRIDE           = [$BIT],        ! Override master index information
        NDX$V_STANDARD_PAGE      = [$BIT],        ! Generate standard page numbers
        NDX$V_CONTINUATION       = [$BIT],        ! Generate continuation headings
        NDX$V_GUIDE              = [$BIT],        ! Generate guide headings
        NDX$V_WORD_SORT          = [$BIT],        ! Sort entries word by word
        NDX$V_LOG                = [$BIT],        ! Generate /LOG message
        NDX$V_MASTER             = [$BIT],        ! Generate a master index
        NDX$V_PAGE_MERGE         = [$BIT],        ! Merge adjacent page references
        NDX$V_TELLTALE           = [$BIT],        ! Generate telltale headings

        $CONTINUE

    NDX$H_FORMAT         = [$SHORT_INTEGER],     ! Output format: DSR, TMS, TEX
    NDX$H_LAYOUT         = [$SHORT_INTEGER],     ! Output layout type
    NDX$H_NONALPHA       = [$SHORT_INTEGER],     ! Treatment of leading nonalphas during sort
    NDX$H_LEVEL          = [$SHORT_INTEGER],     ! Deepest level to include in index
    NDX$G_COLUMN_WID     = [$INTEGER],           ! Column width
    NDX$G_GUTTER_WID     = [$INTEGER],           ! Gutter width
    NDX$G_LINES_PAGE     = [$INTEGER],           ! Lines per page
    NDX$G_RESERVE_LINES = [$INTEGER],            ! Number of lines to reserve when requiring a file
    NDX$G_SEPARATE_WIDTH= [$INTEGER],            ! Width of reference portion of entry
    NDX$T_MASTER_BOOK    = [$DESCRIPTOR(DYNAMIC)],  ! Book name descriptor for Master indexing
    NDX$T_INPUT_FILE     = [$DESCRIPTOR(DYNAMIC)],  ! Input file name descriptor
    NDX$T_OUTPUT_FILE    = [$DESCRIPTOR(DYNAMIC)],  ! Output file name descriptor
    NDX$T_REQUIRE_FILE   = [$DESCRIPTOR(DYNAMIC)],  ! Require file name descriptor
    NDX$T_RELATED_FILE   = [$DESCRIPTOR(DYNAMIC)],  ! Related file name descriptor is saved here
                                                    ! by NDXINP for later use by MAKNDX
    NDX$T_COMMAND_LINE   = [$DESCRIPTOR(DYNAMIC)]   ! Copy of entire command line

    TES;
!
!   End of NDXCMD_FIELDS
!

LITERAL
    NDXCMD$K_LENGTH = $FIELD_SET_SIZE;

MACRO
    $NDXCMD = BLOCK [NDXCMD$K_LENGTH] FIELD (NDXCMD_FIELDS) %;

$LITERAL                                    ! Output formats (NDX$H_FORMAT)
    DSR                      = $DISTINCT,   ! Runoff
    TMS11_A                  = $DISTINCT,   ! TMS=A
```

```
;   R0262  1          TMS11_E                   = $DISTINCT,      ! TMS=E
;   R0263  1          TEX                       = $DISTINCT;      ! TEX
;   R0264  1
;   R0265  1      $LITERAL                                        ! Output layouts (NDX$H_LAYOUT)
;   R0266  1          TWO_COLUMN                = $DISTINCT,      ! Normal two column format
;   R0267  1          ONE_COLUMN                = $DISTINCT,      ! Normal one column format
;   R0268  1          SEPARATE                  = $DISTINCT,      ! Separate reference format
;   R0269  1          GALLEY                    = $DISTINCT;      ! TMS11 Galley format
;   R0270  1
;   R0271  1      $LITERAL                                        ! Treatment of leading nonalphas during sort (NDX$H_NONALPHA)
;   R0272  1          BEFORE                    = $DISTINCT,      ! Leading nonalphas sort before alphas
;   R0273  1          AFTER                     = $DISTINCT,      ! Leading nonalphas sort after alphas
;   R0274  1          IGNORE                    = $DISTINCT;      ! Leading nonalphas are ignored
;   R0275  1
;   R0276  1      !
;   R0277  1      !--      End of NDXCLI.REQ
;  138          0278  1
;  139          0279  1 REQUIRE 'REQ:NDXVMSREQ';                        ! Error message definitions
```

```
!
! Version:       'V04-000'
!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!


!++
! FACILITY:
!   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
!
! ABSTRACT:
!       This file contains external references to the error message numbers
!       for DSRINDEX/INDEX.
!
!       New messages must be defined in NDXVMSMSG.MSG and referenced here:
!       both in the MACRO section (for DSRINDEX) and the EXTERNAL LITERAL
!       section (for INDEX)
!
! ENVIRONMENT:   VAX/VMS User Mode
!
! AUTHOR:        JPK
!
! CREATION DATE: 01-Feb-1983
!
! MODIFIED BY:
!
!       004     JPK00022        30-Mar-1983
!               Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
!               to generate TEX output. Added module NDXTEX.
!
!       003     JPK00021        28-Mar-1983
!               Modified NDXT20 to include E2.0 functionality.
!               Modified NDXCLIDMP, NDXFMT, NDXPAG, NDXVRS to require RNODEF
!               for BLISS36 and to remove any conditional require based on
!               DSRPLUS_DEF.
!
```

```
;  R0337  1        !         002     JPK00010        04-Feb-1983
;  R0338  1        !                 Cleaned up module names, modified revision history to
;  R0339  1        !                 conform with established standards. Updated copyright dates.
;  R0340  1        !
;  R0341  1        !--
;  R0342  1
;  R0343  1        REQUIRE 'REQ:RNODEF';
```

```
!
! Version:        'V04-000'
!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!


!++
! FACILITY:       DSR (Digital Standard RUNOFF) / DSRPLUS
!
! ABSTRACT:
!      Converts BLISS/VARIANT values into useful names.
!
! ENVIRONMENT:    Transportable BLISS
!
! AUTHOR:         Rich Friday
!
! CREATION DATE: 1978
!
! MODIFIED BY:
!
!      016      KAD00016        Ray Marshall    19-Mar-1984
!               Added GERMAN, FRENCH, & ITALIAN.
!
!      015      KAD00015        Keith Dawson    18-Apr-1983
!               Made the LN01 conditional the default for vanilla DSR --
!                 its value is 0 (no variant supplied).
!
!      014      KAD00014        Keith Dawson    22-Mar-1983
!               Asserted the LN01 conditional when DSRPLUS is asserted.
!
!      013      KAD00013        Keith Dawson    20-Mar-1983
!               Removed all references to .BIX and .BTC files.
!
!      012      KAD00012        Keith Dawson    07-Mar-1983
!               Global edit of all modules. Updated module names, idents,
!               copyright dates. Changed require files to BLISS library.
```

```
;   1   !
;   1   !--
;   1
;   1   !++
;   1   !       D E F I N I T I O N   O F  /VARIANT   B I T S
;   1   !
;   1   !       The bit assignments are as follows:
;   1   !
;   1   !       Bit  Weight   Meaning
;   1   !-------------------------------------------------------------------------------
;   1   !       --      0     If no /VARIANT is supplied (as for vanilla DSR),
;   1   !                     compile with LN01 support. LN01 support is also
;   1   !                     implied by the DSRPLUS variant.
;   1   !
;   1   !        0      1     CLEAR =   Unassigned
;   1   !                     SET   =   Unassigned
;   1   !
;   1   !        1      2     CLEAR =   Normal compile
;   1   !                     SET   =   Compile for DSRPLUS
;   1   !
;   1   !       4-6    16     CLEAR =   English (American) version
;   1   !                     SET   =   16 = German (Austrian)
;   1   !                               32 = French
;   1   !                               48 = Italian
;   1   !--
;   1
;   1   !-----------------------------------------------------------------------------
;   1   !    This variable (LN01) controls whether or not to compile an LN01-flavored
;   1   !    DSR. It is asserted by default, and also whenever DSRPLUS is asserted.
;   1   !
;   1   !    Modules utilizing LN01 are:
;   1   !
;   1   !       DOOPTS  NOUT
;   1
;   1   COMPILETIME
;   1       ln01 =
;   2           ( (%VARIANT EQL 0) OR  %VARIANT/2 )
;   1       ;
;   1
;   1   !-----------------------------------------------------------------------------
;   1   !    This variable (DSRPLUS) controls compilation for the DSRPLUS program.
;   1   !
;   1   !    All modules utilize DSRPLUS.
;   1
;   1   COMPILETIME
;   1       dsrplus =
;   2           ( %VARIANT/2 )
;   1       ;
;   1
;   1   !-----------------------------------------------------------------------------
;   1   !    This variable (FLIP) controls compilation of FLIP features of DSRPLUS.
;   1   !    It assures that FLIP features are compiled only on VMS systems.
;   1   !
;   1   !    Modules utilizing FLIP are many and various.
;   1
;   1   COMPILETIME
;   1       flip =
;  R0458  2              ( %VARIANT/2 AND %BLISS(BLISS32) )
;  R0459  1          ;
;  R0460  1
;  R0461  1      !---------------------------------------------------------------------------------------------
;  R0462  1      !        4-6     16        CLEAR =     English (American) version
;  R0463  1      !                          SET   =     16 = German (Austrian)
;  R0464  1      !                                      32 = French
;  R0465  1      !                                      48 = Italian
;  R0466  1      COMPILETIME
;  R0467  1          German  = ( %VARIANT/16 AND NOT %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0468  1      COMPILETIME
;  R0469  1          French  = ( NOT %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0470  1      COMPILETIME
;  R0471  1          Italian = ( %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0472  1      !---------------------------------------------------------------------------------------------
;  R0473  1      !                       End of RNODEF.REQ
```

```
%IF NOT DSRPLUS
%THEN

MACRO
    INDEX$_BADLOGIC  = DSRINDEX$_BADLOGIC  %,
    INDEX$_BADVALUE  = DSRINDEX$_BADVALUE  %,
    INDEX$_INSVIRMEM = DSRINDEX$_INSVIRMEM %,
    INDEX$_LINELENG  = DSRINDEX$_LINELENG  %,
    INDEX$_NOREF     = DSRINDEX$_NOREF     %,
    INDEX$_OPENIN    = DSRINDEX$_OPENIN    %,
    INDEX$_OPENOUT   = DSRINDEX$_OPENOUT   %,
    INDEX$_TOOMANY   = DSRINDEX$_TOOMANY   %,
    INDEX$_VALERR    = DSRINDEX$_VALERR    %,
    INDEX$_CANTBAL   = DSRINDEX$_CANTBAL   %,
    INDEX$_CLOSEQUOT = DSRINDEX$_CLOSEQUOT %,
    INDEX$_CONFQUAL  = DSRINDEX$_CONFQUAL  %,
    INDEX$_CTRLCHAR  = DSRINDEX$_CTRLCHAR  %,
    INDEX$_DOESNTFIT = DSRINDEX$_DOESNTFIT %,
    INDEX$_DUPBEGIN  = DSRINDEX$_DUPBEGIN  %,
    INDEX$_EMPTYIN   = DSRINDEX$_EMPTYIN   %,
    INDEX$_IGNORED   = DSRINDEX$_IGNORED   %,
    INDEX$_INVINPUT  = DSRINDEX$_INVINPUT  %,
    INDEX$_INVRECORD = DSRINDEX$_INVRECORD %,
    INDEX$_LASTCONT  = DSRINDEX$_LASTCONT  %,
    INDEX$_NOBEGIN   = DSRINDEX$_NOBEGIN   %,
    INDEX$_NOEND     = DSRINDEX$_NOEND     %,
    INDEX$_NOINDEX   = DSRINDEX$_NOINDEX   %,
    INDEX$_NOLIST    = DSRINDEX$_NOLIST    %,
    INDEX$_OVERSTRK  = DSRINDEX$_OVERSTRK  %,
    INDEX$_SKIPPED   = DSRINDEX$_SKIPPED   %,
    INDEX$_SYNTAX    = DSRINDEX$_SYNTAX    %,
    INDEX$_TEXFILE   = DSRINDEX$_TEXFILE   %,
    INDEX$_TOODEEP   = DSRINDEX$_TOODEEP   %,
    INDEX$_TOOFEW    = DSRINDEX$_TOOFEW    %,
    INDEX$_TRUNCATED = DSRINDEX$_TRUNCATED %,
    INDEX$_COMPLETE  = DSRINDEX$_COMPLETE  %,
    INDEX$_CREATED   = DSRINDEX$_CREATED   %,
    INDEX$_IDENT     = DSRINDEX$_IDENT     %,
    INDEX$_PROCFILE  = DSRINDEX$_PROCFILE  %,
    INDEX$_TEXT      = DSRINDEX$_TEXT      %,
    INDEX$_TEXTD     = DSRINDEX$_TEXTD     %,
    INDEX$_TMS11     = DSRINDEX$_TMS11     %;

%FI

EXTERNAL LITERAL
    INDEX$_BADLOGIC,      ! <internal logic error detected>
    INDEX$_BADVALUE,      ! <'!AS' is an invalid keyword value>
    INDEX$_INSVIRMEM,     ! <insufficient virtual memory>
    INDEX$_LINELENG,      ! <maximum line length is 120>
    INDEX$_NOREF,         ! <page reference not found>
    INDEX$_OPENIN,        ! <error opening '!AS' for input>
    INDEX$_OPENOUT,       ! <error opening '!AS' for output>
    INDEX$_TOOMANY,       ! <too many values supplied>
    INDEX$_VALERR,        ! <specified value is out of legal range>
    INDEX$_CANTBAL,       ! <can't balance last page>
```

```
        INDEX$_CLOSEQUOT,     ! <missing close quote>
        INDEX$_CONFQUAL,      ! <conflicting qualifiers>
        INDEX$_CTRLCHAR,      ! <the following line contains control characters - ignored>
        INDEX$_DOESNTFIT,     ! <'!AD' will not fit at the current indentation level>
        INDEX$_DUPBEGIN,      ! <duplicate .XPLUS (BEGIN) - inserted as .XPLUS ()>
        INDEX$_EMPTYIN,       ! <empty input file '!AS'>
        INDEX$_IGNORED,       ! <'!AS' ignored>
        INDEX$_INVINPUT,      ! <invalid input file format in file '!AS'>
        INDEX$_INVRECORD,     ! <invalid record type in file '!AS'>
        INDEX$_LASTCONT,      ! <can't generate continuation heading on last page>
        INDEX$_NOBEGIN,       ! <.XPLUS (END) with no .XPLUS (BEGIN) - inserted as .XPLUS ()>
        INDEX$_NOEND,         ! <.XPLUS (BEGIN) has no corresponding .XPLUS (END)>
        INDEX$_NOINDEX,       ! <no index information in file '!AS'>
        INDEX$_NOLIST,        ! <parameter list not allowed>
        INDEX$_OVERSTRK,      ! <the following line contains an overstrike sequence>
        INDEX$_SKIPPED,       ! <!UL reference!%S inside page range - ignored>
        INDEX$_SYNTAX,        ! <error parsing '!AS'>
        INDEX$_TEXFILE,       ! <error processing line !UL of TEX character file '!AS'>
        INDEX$_TOODEEP,       ! <maximum subindex depth exceeded>
        INDEX$_TOOFEW,        ! <not enough values supplied>
        INDEX$_TRUNCATED,     ! <string too long - truncated>
        INDEX$_COMPLETE,      ! <processing complete '!AS'>
        INDEX$_CREATED,       ! <'!AS' created>
        INDEX$_IDENT,         ! <INDEX version !AD>
        INDEX$_PROCFILE,      ! <processing file '!AS'>
        INDEX$_TEXT,          ! <!AS>
        INDEX$_TEXTD,         ! <entry text: '!AD'>
        INDEX$_TMS11;         ! <output file full - continuing with file '!AS'>
```

```
;  140          0560    1
;  141          0561    1 SWITCHES LIST (NOREQUIRE);
;  142          0562    1
;  143          0563    1 !
;  144          0564    1 ! TABLE OF CONTENTS:
;  145          0565    1 !
;  146          0566    1
;  147          0567    1 FORWARD ROUTINE
;  148          0568    1     NDXCLI,                         ! Command line interface
;  149          0569    1     CONDITION_HANDLER,              ! Main program condition handler - sets return status
;  150          0570    1     CALL_TPARSE,                    ! Invoke TPARSE to parse qualifier values
;  151          0571    1     ENTER_PAGE,                     ! Action routine - enter page number type
;  152          0572    1     OPEN_ERROR;                     ! Report file open errors
;  153          0573    1
;  154        L 0574    1 %IF DSRPLUS
;  155        U 0575    1 %THEN
;  156        U 0576    1
;  157        U 0577    1 FORWARD ROUTINE
;  158        U 0578    1     ENTER_MERGE,                    ! Action routine - enter page range merge
;  159        U 0579    1     ENTER_LAYOUT,                   ! Action routine - enter /LAYOUT value
;  160        U 0580    1     ENTER_FORMAT,                   ! Action routine - enter /FORMAT value
;  161        U 0581    1     ENTER_SORT,                     ! Action routine - enter sort type
;  162        U 0582    1     ENTER_ALPHA,                    ! Action routine - enter nonalpha sort
;  163        U 0583    1     OPTIONS_FILE : NOVALUE,         ! Process options file
;  164        U 0584    1     PARSE_BOOK   : NOVALUE,         ! Process /BOOK_IDENTIFIER qualifier
;  165        U 0585    1     PROCESS_TEX_FILE : NOVALUE,     ! Process TEX character size file
;  166        U 0586    1     STORE_TEX,                      ! Action routine - store TEX character size
;  167        U 0587    1     READ_TEX;                       ! Action routine - read a line from TEX char file
;  168        U 0588    1
;  169          0589    1 %FI
;  170          0590    1
;  171          0591    1 !
;  172          0592    1 ! EQUATED SYMBOLS:
;  173          0593    1 !
;  174          0594    1
;  175          0595    1 LITERAL
;  176          0596    1     TRUE  = 1,
;  177          0597    1     FALSE = 0;
;  178          0598    1
;  179          0599    1 !
;  180          0600    1 ! OWN STORAGE:
;  181          0601    1 !
;  182          0602    1
;  183          0603    1 OWN
;  184          0604    1     VALUE_STR   : $STR_DESCRIPTOR (CLASS = DYNAMIC, STRING = (0, 0)),
;  185          0605    1     OPTIONS_STR : $STR_DESCRIPTOR (CLASS = DYNAMIC, STRING = (0, 0)),
;  186          0606    1     QUALIFIER_VALUE,
;  187          0607    1     TERMINATION_STATUS : INITIAL (STS$K_SUCCESS);
;  188          0608    1
;  189        L 0609    1 %IF DSRPLUS
;  190        U 0610    1 %THEN
;  191        U 0611    1
;  192        U 0612    1 OWN
;  193        U 0613    1     TEX_FILE_NAME : $STR_DESCRIPTOR (CLASS = DYNAMIC, STRING = (0, 0)),
;  194        U 0614    1     TEX_CHAR_SIZES : VECTOR [256],                  ! Where character sizes are stroed
;  195        U 0615    1     TEX_CHAR_INDEX,                                 ! Index into TEX_CHAR_SIZES
;  196        U 0616    1     TEX_FILE_LINE_NO,                               ! Line number of file
;  197    U 0617 1      TEX_LINE : $STR_DESCRIPTOR (),                  ! Descriptor of input line
;  198    U 0618 1      TEX_IN_BUF : BLOCK [512, BYTE],                 ! Input buffer
;  199    U 0619 1      TEX_ES : BLOCK [NAM$C_MAXRSS, BYTE],            ! Expanded filename string
;  200    U 0620 1      TEX_RS : BLOCK [NAM$C_MAXRSS, BYTE],            ! Resultant filename string
;  201    U 0621 1      TEX_NAM : $NAM (ESA = TEX_ES, ESS = NAM$C_MAXRSS, RSA = TEX_RS, RSS = NAM$C_MAXRSS),
;  202    U 0622 1      TEX_FAB : $FAB (NAM = TEX_NAM, DNM = '.FSZ'),
;  203    U 0623 1      TEX_RAB : $RAB (FAB = TEX_FAB, UBF = TEX_IN_BUF, USZ = 512);
;  204    U 0624 1
;  205      0625 1 %FI
;  206      0626 1
;  207      0627 1 !
;  208      0628 1 ! EXTERNAL REFERENCES:
;  209      0629 1 !
;  210      0630 1
;  211      0631 1 EXTERNAL LITERAL
;  212      0632 1      TAB : UNSIGNED (8),         ! TAB character
;  213      0633 1      TMSCOL,                     ! Default TMS column width
;  214      0634 1      MAXLIN;                     ! Maximum number of lines per page
;  215      0635 1
;  216      0636 1 EXTERNAL LITERAL                 ! Values returned from CLI interface
;  217      0637 1      CLI$_CONCAT,                ! Value concatenated to next
;  218      0638 1      CLI$_PRESENT,               ! Value explicitly given
;  219      0639 1      CLI$_NEGATED,               ! Value explicitly negated (/NO)
;  220      0640 1      CLI$_DEFAULTED,             ! Value defaulted present
;  221      0641 1      CLI$_ABSENT;                ! Value not present
;  222      0642 1
;  223      0643 1 EXTERNAL
;  224      0644 1      CMDBLK : $NDXCMD,           ! Command line information block
;  225      0645 1      CHRSIZ : REF VECTOR,        ! TMS character size vector pointer
;  226      0646 1      CHRSZA : VECTOR,            ! Character size vector for /TMS11 = A
;  227      0647 1      CHRSZE : VECTOR,            ! Character size vector for /TMS11 = E
;  228      0648 1      NDXVRL,                     ! Length of version number string
;  229      0649 1      NDXVRP;                     ! CH$PTR to version number string
;  230      0650 1
;  231      0651 1 EXTERNAL ROUTINE
;  232      0652 1      NDXINI          : NOVALUE,                          ! Once only initialization
;  233      0653 1      NDXINP          : NOVALUE,                          ! Process input file
;  234      0654 1      MAKNDX          : NOVALUE,                          ! Generate index
;  235      0655 1      CLI$PRESENT     : ADDRESSING_MODE (GENERAL),        ! Check for qualifier
;  236      0656 1      CLI$GET_VALUE   : ADDRESSING_MODE (GENERAL),        ! Get value of qualifier
;  237      0657 1      LIB$TPARSE      : ADDRESSING_MODE (GENERAL);        ! Table driven parser
;  238      0658 1
;  239    L 0659 1 %IF DSRPLUS
;  240    U 0660 1 %THEN
;  241    U 0661 1
;  242    U 0662 1 EXTERNAL
;  243    U 0663 1      NDXOPTION;                  ! Options file parse tables address
;  244    U 0664 1
;  245    U 0665 1 EXTERNAL ROUTINE
;  246    U 0666 1      CLI$DCL_PARSE   : ADDRESSING_MODE (GENERAL);        ! Initiate new parse
;  247    U 0667 1
;  248      0668 1 %FI
;  249      0669 1
;  250      0670 1 !+
;  251      0671 1 !    TPARSE state tables
;  252      0672 1 !-
;  253      0673 1
;   254          0674  1 !
;   255          0675  1 ! Tables to parse an arbitrary number
;   256          0676  1 !
;   257          0677  1 $INIT_STATE (NUMBER_STATE, NUMBER_KEY);
;   258        P 0678  1 $STATE (,
;   259        P 0679  1     (TPA$_DECIMAL, ,              ,              , QUALIFIER_VALUE),
;   260        P 0680  1     (TPA$_EOS,     TPA$_EXIT)
;   261          0681  1     );
;   262        P 0682  1 $STATE (,
;   263        P 0683  1     (TPA$_EOS,     TPA$_EXIT)
;   264          0684  1     );
;   265          0685  1
;   266          0686  1 !
;   267          0687  1 ! Tables to parse /PAGE_NUMBERS values
;   268          0688  1 !
;   269          0689  1 $INIT_STATE (PAGE_STATE, PAGE_KEY);
;   270        P 0690  1 $STATE (,
;   271        P 0691  1
;   272        L 0692  1 %IF DSRPLUS
;   273        U 0693  1 %THEN
;   274        U 0694  1
;   275        U 0695  1     ('MERGE',     ,             ENTER_MERGE,  , ,               TRUE),
;   276        U 0696  1     ('NOMERGE',   ,             ENTER_MERGE,  , ,               FALSE),
;   277        U 0697  1     ('STANDARD',  ,             ENTER_PAGE,   , ,               TRUE),
;   278        U 0698  1
;   279        P 0699  1 %FI
;   280        P 0700  1
;   281        P 0701  1     ('RUNNING',   ,             ENTER_PAGE,   , ,               FALSE),
;   282        P 0702  1     ('NORUNNING', ,             ENTER_PAGE,   , ,               TRUE)
;   283          0703  1     );
;   284        P 0704  1 $STATE (,
;   285        P 0705  1     (TPA$_EOS,     TPA$_EXIT)
;   286          0706  1     );
;   287          0707  1
;   288        L 0708  1 %IF DSRPLUS
;   289        U 0709  1 %THEN
;   290        U 0710  1
;   291        U 0711  1 !
;   292        U 0712  1 ! Tables to parse /FORMAT values
;   293        U 0713  1 !
;   294        U 0714  1 $INIT_STATE (FORMAT_STATE, FORMAT_KEY);
;   295        U 0715  1 $STATE (,
;   296        U 0716  1     ('DSR',        FORMAT_END, ENTER_FORMAT, , ,               DSR),
;   297        U 0717  1     ('TEX',        TEX_STATE),
;   298        U 0718  1     ('TMS',        TMS_STATE)
;   299        U 0719  1     );
;   300        U 0720  1 $STATE (TEX_STATE,
;   301        U 0721  1     ('=',          TPA$_EXIT,  ENTER_FORMAT, , ,               TEX),
;   302        U 0722  1     (':',          TPA$_EXIT,  ENTER_FORMAT, , ,               TEX)
;   303        U 0723  1     );
;   304        U 0724  1 $STATE (TMS_STATE,
;   305        U 0725  1     ('='),
;   306        U 0726  1     (':'),
;   307        U 0727  1     (TPA$_EOS,     TPA$_EXIT,  ENTER_FORMAT, , ,               TMS11_A)
;   308        U 0728  1     );
;   309        U 0729  1 $STATE (,
;   310        U 0730  1     ('A',          FORMAT_END, ENTER_FORMAT, , ,               TMS11_A),
;   311        U 0731  1       ('E',           FORMAT_END, ENTER_FORMAT, , ,                       TMS11_E)
;   312        U 0732  1       );
;   313        U 0733  1   $STATE (FORMAT_END,
;   314        U 0734  1       (TPA$_EOS,      TPA$_EXIT)
;   315        U 0735  1       );
;   316        U 0736  1
;   317        U 0737  1   !
;   318        U 0738  1   ! Tables to parse /LAYOUT values
;   319        U 0739  1   !
;   320        U 0740  1   $INIT_STATE (LAYOUT_STATE, LAYOUT_KEY);
;   321        U 0741  1   $STATE (,
;   322        U 0742  1       ('TWO_COLUMN', LAYOUT_END, ENTER_LAYOUT, , ,                        TWO_COLUMN),
;   323        U 0743  1       ('2',          LAYOUT_END, ENTER_LAYOUT, , ,                        TWO_COLUMN),
;   324        U 0744  1       ('ONE_COLUMN', LAYOUT_END, ENTER_LAYOUT, , ,                        ONE_COLUMN),
;   325        U 0745  1       ('1',          LAYOUT_END, ENTER_LAYOUT, , ,                        ONE_COLUMN),
;   326        U 0746  1       ('GALLEY',     LAYOUT_END, ENTER_LAYOUT, , ,                        GALLEY),
;   327        U 0747  1       ('SEPARATE',   ,           ENTER_LAYOUT, , ,                        SEPARATE)
;   328        U 0748  1       );
;   329        U 0749  1   $STATE (,
;   330        U 0750  1       ('='),
;   331        U 0751  1       (':'),
;   332        U 0752  1       (TPA$_EOS,     TPA$_EXIT)
;   333        U 0753  1       );
;   334        U 0754  1   $STATE (,
;   335        U 0755  1       (TPA$_DECIMAL, LAYOUT_END, ,                    , QUALIFIER_VALUE)
;   336        U 0756  1       );
;   337        U 0757  1   $STATE (LAYOUT_END,
;   338        U 0758  1       (TPA$_EOS,     TPA$_EXIT)
;   339        U 0759  1       );
;   340        U 0760  1
;   341        U 0761  1   !
;   342        U 0762  1   ! Tables to parse /SORT values
;   343        U 0763  1   !
;   344        U 0764  1   $INIT_STATE (SORT_STATE, SORT_KEY);
;   345        U 0765  1   $STATE (,
;   346        U 0766  1       ('WORD',       SORT_END,   ENTER_SORT,   , ,                        TRUE),
;   347        U 0767  1       ('LETTER',     SORT_END,   ENTER_SORT,   , ,                        FALSE),
;   348        U 0768  1       ('NONALPHA')
;   349        U 0769  1       );
;   350        U 0770  1   $STATE (,
;   351        U 0771  1       ('='),
;   352        U 0772  1       (':')
;   353        U 0773  1       );
;   354        U 0774  1   $STATE (,
;   355        U 0775  1       ('IGNORE',     SORT_END,   ENTER_ALPHA,  , ,                        IGNORE),
;   356        U 0776  1       ('BEFORE',     SORT_END,   ENTER_ALPHA,  , ,                        BEFORE),
;   357        U 0777  1       ('AFTER',      SORT_END,   ENTER_ALPHA,  , ,                        AFTER)
;   358        U 0778  1       );
;   359        U 0779  1   $STATE (SORT_END,
;   360        U 0780  1       (TPA$_EOS,     TPA$_EXIT)
;   361        U 0781  1       );
;   362        U 0782  1
;   363        U 0783  1   !
;   364        U 0784  1   ! Tables to parse TEX character size file
;   365        U 0785  1   !
;   366        U 0786  1   $INIT_STATE (TEX_FILE_STATE, TEX_FILE_KEY);
;   367        U 0787  1   $STATE (TEX_1,
;   368         U 0788  1       ('!',           TEX_1,          READ_TEX),
;   369         U 0789  1       (TPA$_EOS,      TEX_1,          READ_TEX),
;   370         U 0790  1       (TPA$_DECIMAL,  TEX_2,          STORE_TEX)
;   371         U 0791  1       );
;   372         U 0792  1 $STATE (TEX_2,
;   373         U 0793  1       (',',           TEX_1),
;   374         U 0794  1       ('!',           TEX_2,          READ_TEX),
;   375         U 0795  1       (TPA$_EOS,      TEX_2,          READ_TEX)
;   376         U 0796  1       );
;   377         U 0797  1
;   378           0798  1 %FI
```

```
1 %SBTTL 'NDXCLI -- Main program - command line interface'
1 GLOBAL ROUTINE NDXCLI =
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine uses the VMS DCL CLE to obtain command
1 !       line information which is in turn passed to the INDEX
1 !       application in a transportable manner.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       CMDBLK - The command line information block is filled in
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       TERMINATION_STATUS      - Set by CONDITION_HANDLER ()
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     ENABLE
2         CONDITION_HANDLER;
2
2     LOCAL
2         STATUS;
2
2     NDXINI ();                          ! Do once-only initialization
2
2     !
2     ! Get copy of whole command line
2     !
2     CLI$GET_VALUE (%ASCID'$LINE', CMDBLK [NDX$T_COMMAND_LINE]);
2
2     !
2     ! /[NO]MASTER
2     !
2     !                   * W A R N I N G *
2     !
2     !           This must be parsed before other qualifiers.
2     !           Other qualifiers depend on the value of this qualifier.
2     !
2     !                   * W A R N I N G *
```

```
;  437         0856  2          !
;  438         0857  2          CMDBLK [NDX$V_MASTER] = FALSE;
;  439         0858  2
;  440       L 0859  2  %IF DSRPLUS
;  441       U 0860  2  %THEN
;  442       U 0861  2
;  443       U 0862  2          IF  CLI$PRESENT (%ASCID'MASTER')
;  444       U 0863  2          THEN
;  445       U 0864  2              CMDBLK [NDX$V_MASTER] = TRUE;
;  446         0865  2  %FI
;  447         0866  2
;  448         0867  2          !
;  449         0868  2          ! /FORMAT = { DSR | TEX : filename | TMS11 [ = { A | E }] }
;  450         0869  2          !
;  451         0870  2          !                    * W A R N I N G *
;  452         0871  2          !
;  453         0872  2          !         This must be parsed before other qualifiers.
;  454         0873  2          !         Other qualifiers depend on the value of this qualifier.
;  455         0874  2          !
;  456         0875  2          !                    * W A R N I N G *
;  457         0876  2          !
;  458         0877  2          CMDBLK [NDX$H_FORMAT] = DSR;          ! Assume output for RUNOFF
;  459         0878  2          CHRSIZ                = CHRSZA;       ! Assume TMS11 type 'A' characters
;  460         0879  2
;  461       L 0880  2  %IF DSRPLUS
;  462       U 0881  2  %THEN
;  463       U 0882  2
;  464       U 0883  2          IF  CLI$PRESENT (%ASCID'FORMAT')
;  465       U 0884  2          THEN
;  466       U 0885  2              BEGIN
;  467       U 0886  2              CLI$GET_VALUE (%ASCID'FORMAT', VALUE_STR);
;  468       U 0887  2
;  469       U 0888  2              IF  NOT CALL_TPARSE (VALUE_STR, FORMAT_STATE, FORMAT_KEY)
;  470       U 0889  2              THEN
;  471       U 0890  2                  SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  472       U 0891  2
;  473       U 0892  2              IF .CMDBLK [NDX$H_FORMAT] EQL TEX THEN PROCESS_TEX_FILE ();
;  474       U 0893  2              END;
;  475       U 0894  2
;  476         0895  2  %FI
;  477         0896  2
;  478         0897  2          !
;  479         0898  2          ! /COLUMN_WIDTH = n
;  480         0899  2          !
;  481         0900  2          !                    * W A R N I N G *
;  482         0901  2          !
;  483         0902  2          !         This must be parsed after /FORMAT and before any other
;  484         0903  2          !         qualifier. It depends on the value of /FORMAT and other
;  485         0904  2          !         qualifiers depend on the value of this qualifier.
;  486         0905  2          !
;  487         0906  2          !                    * W A R N I N G *
;  488         0907  2          !
;  489         0908  2          CMDBLK [NDX$G_COLUMN_WID] = 34;                   ! Default column width is 34
;  490         0909  2
;  491       L 0910  2  %IF DSRPLUS
;  492       U 0911  2  %THEN
;  493       U 0912  2
```

```
U 2       IF  CLI$PRESENT (%ASCID'COLUMN_WIDTH')
U 2       THEN
U 2           BEGIN
U 2           QUALIFIER_VALUE = 0;
U 2           CLI$GET_VALUE (%ASCID'COLUMN_WIDTH', VALUE_STR);
U 2
U 2           IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
U 2           THEN
U 2               SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
U 2
U 2           CMDBLK [NDX$G_COLUMN_WID] = .QUALIFIER_VALUE;
U 2
U 2           IF  .CMDBLK [NDX$G_COLUMN_WID] LSS 5
U 2           THEN
U 2               SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
U 2
U 2           END
U 2       ELSE
U 2           BEGIN
U 2
U 2           IF .CMDBLK [NDX$H_FORMAT] NEQ DSR
U 2           THEN
U 2               !
U 2               ! Typeset column width default is defined by the literal TMSCOL
U 2               !
U 2               CMDBLK [NDX$G_COLUMN_WID] = TMSCOL;
U 2
U 2           END;
U 2
  2   %FI
  2
  2       !
  2       ! /LAYOUT = { TWO_COLUMN | ONE_COLUMN | GALLEY | SEPARATE [= n] }
  2       !
  2       !                     * W A R N I N G *
  2       !
  2       !         This must be parsed after /COLUMN_WIDTH and before any other
  2       !         qualifier. It depends on the value of /COLUMN_WIDTH and other
  2       !         qualifiers depend on the value of this qualifier.
  2       !
  2       !                     * W A R N I N G *
  2       !
  2       CMDBLK [NDX$H_LAYOUT]         = TWO_COLUMN; ! Default index layout
  2       CMDBLK [NDX$G_SEPARATE_WIDTH] = .CMDBLK [NDX$G_COLUMN_WID];
  2
L 2   %IF DSRPLUS
U 2   %THEN
U 2
U 2       IF  CLI$PRESENT (%ASCID'LAYOUT')
U 2       THEN
U 2           BEGIN
U 2           QUALIFIER_VALUE = -1;
U 2           CLI$GET_VALUE (%ASCID'LAYOUT', VALUE_STR);
U 2
U 2           IF  NOT CALL_TPARSE (VALUE_STR, LAYOUT_STATE, LAYOUT_KEY)
U 2           THEN
U 2               SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
```

```
;  551     U 0970  2
;  552     U 0971  2            IF  .QUALIFIER_VALUE NEQ -1
;  553     U 0972  2            THEN
;  554     U 0973  2                BEGIN
;  555     U 0974  2                !
;  556     U 0975  2                ! Doing SEPARATE index and user specified reference column width.
;  557     U 0976  2                ! Validate against minimum column width.
;  558     U 0977  2                !
;  559     U 0978  2
;  560     U 0979  2                IF  .QUALIFIER_VALUE LSS 5
;  561     U 0980  2                THEN
;  562     U 0981  2                    SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
;  563     U 0982  2
;  564     U 0983  2                CMDBLK [NDX$G_SEPARATE_WIDTH] = .QUALIFIER_VALUE;
;  565     U 0984  2                END;
;  566     U 0985  2
;  567     U 0986  2            END;
;  568     U 0987  2
;  569     U 0988  2        IF  (.CMDBLK [NDX$H_FORMAT] EQL TEX)
;  570     U 0989  2        AND (.CMDBLK [NDX$H_LAYOUT] NEQ TWO_COLUMN)
;  571     U 0990  2        THEN
;  572     U 0991  2            BEGIN
;  573     U 0992  2
;  574     U 0993  2            LOCAL
;  575     U 0994  2                FORMAT_PTR;
;  576     U 0995  2
;  577     U 0996  2            FORMAT_PTR = (
;  578     U 0997  2                SELECTONE .CMDBLK [NDX$H_LAYOUT] OF
;  579     U 0998  2                    SET
;  580     U 0999  2
;  581     U 1000  2                    [ONE_COLUMN]:    %ASCID 'ONE_COLUMN';
;  582     U 1001  2                    [GALLEY]:        %ASCID 'GALLEY';
;  583     U 1002  2                    [SEPARATE]:      %ASCID 'SEPARATE';
;  584     U 1003  2
;  585     U 1004  2                    TES
;  586     U 1005  2                );
;  587     U 1006  2
;  588     U 1007  2            SIGNAL_STOP (INDEX$_BADVALUE, 1, .FORMAT_PTR, INDEX$_CONFQUAL);
;  589     U 1008  2            END;
;  590     U 1009  2
;  591       1010  2 %FI
;  592       1011  2
;  593       1012  2        !
;  594       1013  2        ! /[NO]TELLTALE_HEADINGS
;  595       1014  2        !
;  596       1015  2        !                   * W A R N I N G *
;  597       1016  2        !
;  598       1017  2        !       This must be parsed after /LAYOUT and before /LINES_PER_PAGE
;  599       1018  2        !       It depends on the value of /LAYOUT and /LINES_PER_PAGE
;  600       1019  2        !       depends on the value of this qualifier.
;  601       1020  2        !
;  602       1021  2        !                   * W A R N I N G *
;  603       1022  2        !
;  604       1023  2        CMDBLK [NDX$V_TELLTALE] = FALSE;
;  605       1024  2
;  606     L 1025  2 %IF DSRPLUS
;  607     U 1026  2 %THEN
```

```
        IF  CLI$PRESENT (%ASCID'TELLTALE_HEADINGS')
        THEN
            BEGIN

            IF  .CMDBLK [NDX$H_LAYOUT] EQL GALLEY
            THEN
                !
                ! Doing TMS11 galley output.
                ! Telltale headings are not allowed
                !
                SIGNAL (INDEX$_IGNORED, 1, %ASCID'TELLTALE_HEADINGS', INDEX$_CONFQUAL)
            ELSE
                CMDBLK [NDX$V_TELLTALE] = TRUE;

            END;

%FI

        !
        ! /LINES_PER_PAGE = n
        !
        !                   * W A R N I N G *
        !
        !           This must be parsed after /FORMAT, /LAYOUT and
        !           /TELLTALE_HEADINGS. It depends on the value of these qualifiers
        !
        !                   * W A R N I N G *
        !
        IF .CMDBLK [NDX$H_FORMAT] EQL DSR
        THEN
            BEGIN                                       ! Formatting for RUNOFF

            IF .CMDBLK [NDX$V_TELLTALE]
            THEN
                CMDBLK [NDX$G_LINES_PAGE] = 52          ! 52 lines with /TELLTALE
            ELSE
                CMDBLK [NDX$G_LINES_PAGE] = 55;         ! 55 lines per page otherwise

            END
        ELSE
            CMDBLK [NDX$G_LINES_PAGE] = 54;             ! 54 lines per page for Typeset

        IF  CLI$PRESENT (%ASCID'LINES_PER_PAGE')
        THEN
            BEGIN
            !
            ! User specified a value
            !

            IF  .CMDBLK [NDX$H_LAYOUT] EQL GALLEY
            THEN
                !
                ! Galley output - ignore lines-per-page
                !
                SIGNAL (INDEX$_IGNORED, 1, %ASCID'LINES_PER_PAGE', INDEX$_CONFQUAL)
            ELSE
```

```
  4             BEGIN
  4             QUALIFIER_VALUE = 0;
  4             CLI$GET_VALUE (%ASCID'LINES_PER_PAGE', VALUE_STR);
  4
  4             IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
  4             THEN
  4                 SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
  4
  4             CMDBLK [NDX$G_LINES_PAGE] = .QUALIFIER_VALUE;
  4
  5             IF  (
  6                 (.CMDBLK [NDX$G_LINES_PAGE] LSS 15)
  6                 AND (.CMDBLK [NDX$H_FORMAT] EQL DSR)
  5                 )
  5             OR  (
  6                 (.CMDBLK [NDX$G_LINES_PAGE] LSS 25)
  6                 AND (.CMDBLK [NDX$H_FORMAT] NEQ DSR)
  5                 )
  5             OR  (.CMDBLK [NDX$G_LINES_PAGE] GTR MAXLIN)
  4             THEN
  4                 SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
  4
  3             END;
  3
  2         END;
  2
  2     !
  2     ! /GUTTER_WIDTH = n
  2     !
  2     !                   * W A R N I N G *
  2     !
  2     !       This qualifier depends on the value of /LAYOUT
  2     !
  2     !                   * W A R N I N G *
  2     !
  2     CMDBLK [NDX$G_GUTTER_WID] = 2;
  2
L 2 %IF DSRPLUS
U 2 %THEN
U 2
U 2     IF  (.CMDBLK [NDX$H_LAYOUT] EQL ONE_COLUMN) OR
U 2         (.CMDBLK [NDX$H_LAYOUT] EQL GALLEY)
U 2     THEN
U 2         BEGIN
U 2         !
U 2         ! ONE_COLUMN output which is not a separate master index
U 2         ! or GALLEY output
U 2         !
U 2         CMDBLK [NDX$G_GUTTER_WID] = 0;          ! Gutter width is meaningless
U 2
U 2         IF  CLI$PRESENT (%ASCID'GUTTER_WIDTH')
U 2         THEN
U 2             SIGNAL (INDEX$_IGNORED, 1, %ASCID'GUTTER_WIDTH', INDEX$_CONFQUAL);
U 2
U 2         END
U 2     ELSE
U 2         BEGIN
```

```
    !
    ! For all other page layouts
    !
    QUALIFIER_VALUE = 2;                        ! Default value

    IF  CLI$PRESENT (%ASCID'GUTTER_WIDTH')
    THEN
        BEGIN
        CLI$GET_VALUE (%ASCID'GUTTER_WIDTH', VALUE_STR);

        IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
        THEN
            SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);

        END;

    CMDBLK [NDX$G_GUTTER_WID] = .QUALIFIER_VALUE;
    END;

%FI

  !
  ! Validate the combinations of column width, gutter width, and
  ! right column width for master indexes.
  !
  !                     * W A R N I N G *
  !
  !          This code depends on the value of /LAYOUT, /COLUMN_WIDTH
  !          and /GUTTER_WIDTH
  !
  !                     * W A R N I N G *
  !
  SELECTONE .CMDBLK [NDX$H_LAYOUT] OF
  SET

  [TWO_COLUMN]:

    IF  (2 * .CMDBLK [NDX$G_COLUMN_WID]) +
        .CMDBLK [NDX$G_GUTTER_WID] GTR 120
    THEN
        SIGNAL_STOP (INDEX$_LINELENG);

  [SEPARATE]:

    IF  .CMDBLK [NDX$G_COLUMN_WID] +
        .CMDBLK [NDX$G_GUTTER_WID] +
        .CMDBLK [NDX$G_SEPARATE_WIDTH] GTR 120
    THEN
        SIGNAL_STOP (INDEX$_LINELENG);

  [OTHERWISE]:

    IF  .CMDBLK [NDX$G_COLUMN_WID] GTR 120
    THEN
        SIGNAL_STOP (INDEX$_LINELENG);

  TES;
```

```
        !
        ! /[NO]CONTINUATION_HEADINGS
        !
        !                   * W A R N I N G *
        !
        !           This qualifier depends on the value of /LAYOUT
        !
        !                   * W A R N I N G *
        !
        CMDBLK [NDX$V_CONTINUATION] = FALSE;

%IF DSRPLUS
%THEN

        IF  CLI$PRESENT (%ASCID'CONTINUATION_HEADINGS')
        THEN
            BEGIN

            IF  .CMDBLK [NDX$H_LAYOUT] EQL GALLEY
            THEN
                !
                ! Doing TMS11 galley output.
                ! Continuation headings are not allowed
                !
                SIGNAL (INDEX$_IGNORED, 1, %ASCID'CONTINUATION_HEADINGS', INDEX$_CONFQUAL)
            ELSE
                CMDBLK [NDX$V_CONTINUATION] = TRUE;

            END;

%FI

        !
        ! /NORESERVE
        ! /RESERVE = n
        !
        !                   * W A R N I N G *
        !
        !           This qualifier depends on the value of /LINES_PER_PAGE
        !
        !                   * W A R N I N G *
        !
        CMDBLK [NDX$G_RESERVE_LINES] = 0;

        IF  CLI$PRESENT (%ASCID'RESERVE')
        THEN
            BEGIN
            QUALIFIER_VALUE = 0;
            CLI$GET_VALUE (%ASCID'RESERVE', VALUE_STR);

            IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
            THEN
                SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);

            IF  .QUALIFIER_VALUE GTR .CMDBLK [NDX$G_LINES_PAGE]
            THEN
```

```
   3                 SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
   3
   3             CMDBLK [NDX$G_RESERVE_LINES] = .QUALIFIER_VALUE;
   2             END;
   2
   2         !
   2         ! /LEVEL = n
   2         !
   2         !                       * W A R N I N G *
   2         !
   2         !           This qualifier depends on the value of /MASTER
   2         !
   2         !                       * W A R N I N G *
   2         !
   2         CMDBLK [NDX$H_LEVEL] = 99;                      ! All levels
   2
L  2     %IF DSRPLUS
U  2     %THEN
U  2
U  2         IF  CLI$PRESENT (%ASCID'LEVEL')
U  2         THEN
U  2             BEGIN
U  2             CLI$GET_VALUE (%ASCID'LEVEL', VALUE_STR);
U  2
U  2             IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
U  2             THEN
U  2                 SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
U  2
U  2             IF  .QUALIFIER_VALUE LEQ 0
U  2             THEN
U  2                 SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
U  2
U  2             CMDBLK [NDX$H_LEVEL] = .QUALIFIER_VALUE - 1;
U  2             END
U  2         ELSE
U  2             BEGIN
U  2
U  2             IF  .CMDBLK [NDX$V_MASTER]
U  2             THEN
U  2                 CMDBLK [NDX$H_LEVEL] = 1;           ! Levels 0 and 1 for /MASTER
U  2
U  2             END;
U  2
   2     %FI
   2
   2         !
   2         ! /[NO]GUIDE_HEADINGS
   2         !
   2         CMDBLK [NDX$V_GUIDE] = FALSE;
   2
L  2     %IF DSRPLUS
U  2     %THEN
U  2
U  2         IF  CLI$PRESENT (%ASCID'GUIDE_HEADINGS')
U  2         THEN
U  2             CMDBLK [NDX$V_GUIDE] = TRUE;
U  2
```

```
  2 %FI
  2
  2     !
  2     ! /[NO]IDENTIFICATION
  2     !
  2
  2     IF  CLI$PRESENT (%ASCID'IDENTIFICATION')
  2     THEN
  2         SIGNAL (INDEX$_IDENT, 2, .NDXVRL, .NDXVRP);
  2
  2     !
  2     ! /[NO]LOG
  2     !
  2
  2     IF  CLI$PRESENT (%ASCID'LOG')
  2     THEN
  2         CMDBLK [NDX$V_LOG] = TRUE
  2     ELSE
  2         CMDBLK [NDX$V_LOG] = FALSE;
  2
  2     !
  2     ! /NOOUTPUT
  2     ! /OUTPUT = filespec
  2     !
  2
  2     IF CLI$PRESENT (%ASCID'OUTPUT')
  2     THEN
  3         BEGIN
  3         CMDBLK [NDX$V_OUTPUT] = TRUE;
  3
  3         CLI$GET_VALUE (%ASCID'OUTPUT', CMDBLK [NDX$T_OUTPUT_FILE]);
  3         END
  2     ELSE
  2         CMDBLK [NDX$V_OUTPUT] = FALSE;
  2
  2     !
  2     ! /[NO]OVERRIDE
  2     !
  2     CMDBLK [NDX$V_OVERRIDE] = FALSE;
  2
L 2 %IF DSRPLUS
U 2 %THEN
U 2
U 2     IF  CLI$PRESENT (%ASCID'OVERRIDE_MASTER')
U 2     THEN
U 2         CMDBLK [NDX$V_OVERRIDE] = TRUE;
U 2
  2 %FI
  2
  2     !
  2     ! /NOPAGE_NUMBERS
  2     ! /PAGE_NUMBERS = ([[NO]RUNNING], [[NO]MERGE])
  2     !
  2     ! NORUNNING is the same as STANDARD.
  2     !
  2     CMDBLK [NDX$V_STANDARD_PAGE] = TRUE;        ! Generate standard page numbers
  2     CMDBLK [NDX$V_PAGES]         = TRUE;        ! Generate page numbers
```

```
    2
L   2 %IF NOT DSRPLUS
    2 %THEN
    2
    2      CMDBLK [NDX$V_PAGE_MERGE]    = TRUE;          ! Merge page numbers for DSR
    2
U   2 %ELSE
U   2
U   2      CMDBLK [NDX$V_PAGE_MERGE]    = FALSE;         ! Page ranges formed by .XPLUS (BEGIN - END)
U   2
    2 %FI
    2
    2      SELECTONE CLI$PRESENT (%ASCID'PAGE_NUMBERS') OF
    2          SET
    2
    2          [CLI$_NEGATED]:
    2              !
    2              ! Qualifier explicitly negated (/NOPAGE_NUMBERS).
    2              !
    2              CMDBLK [NDX$V_PAGES] = FALSE;
    2
    2          [CLI$_PRESENT]:
    3              BEGIN
    3              !
    3              ! Qualifier was given explicitly on command line.
    3              !
    3              WHILE CLI$GET_VALUE (%ASCID'PAGE_NUMBERS', VALUE_STR) DO
    3
    3                  IF  NOT CALL_TPARSE (VALUE_STR, PAGE_STATE, PAGE_KEY)
    3                  THEN
    3                      SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
    3
    2              END;
    2
    2          [OTHERWISE]:
    2              !
    2              ! CLI$_ABSENT, CLI$_DEFAULTED.
    2              ! Qualifier is present by default.
    2              !
    2              !
    2              ;
    2
    2          TES;
    2
    2      !
    2      ! /NOREQUIRE
    2      ! /REQUIRE = filespec
    2      !
    2      CMDBLK [NDX$V_REQUIRE] = FALSE;
    2
    2      IF  CLI$PRESENT (%ASCID'REQUIRE')
    2      THEN
    3          BEGIN
    3          CMDBLK [NDX$V_REQUIRE] = TRUE;
    3          CLI$GET_VALUE (%ASCID'REQUIRE', CMDBLK [NDX$T_REQUIRE_FILE]);
    2          END;
    2
    2      !
```

```
; 1007          1426  2         ! /SORT = ([{ WORD | LETTER }], [NONALPHA = { IGNORE | BEFORE | AFTER }])
; 1008          1427  2         !
; 1009          1428  2         CMDBLK [NDX$V_WORD_SORT] = TRUE;                ! Word by word sort is default
; 1010          1429  2         CMDBLK [NDX$H_NONALPHA]  = IGNORE;              ! Ignore leading nonalphas
; 1011          1430  2
; 1012        L 1431  2 %IF DSRPLUS
; 1013        U 1432  2 %THEN
; 1014        U 1433  2
; 1015        U 1434  2         IF  CLI$PRESENT (%ASCID'SORT')
; 1016        U 1435  2         THEN
; 1017        U 1436  2
; 1018        U 1437  2             WHILE CLI$GET_VALUE (%ASCID'SORT', VALUE_STR) DO
; 1019        U 1438  2
; 1020        U 1439  2                 IF  NOT CALL_TPARSE (VALUE_STR, SORT_STATE, SORT_KEY)
; 1021        U 1440  2                 THEN
; 1022        U 1441  2                     SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
; 1023        U 1442  2
; 1024          1443  2 %FI
; 1025          1444  2
; 1026          1445  2         !
; 1027          1446  2         ! Process all input files and local qualifiers
; 1028          1447  2         !
; 1029          1448  2         CMDBLK [NDX$V_INPUT_CONCAT] = FALSE;
; 1030          1449  2
; 1031          1450  2         WHILE (STATUS = CLI$GET_VALUE (%ASCID'INPUT', CMDBLK [NDX$T_INPUT_FILE])) DO
; 1032          1451  3             BEGIN
; 1033          1452  3
; 1034        L 1453  3 %IF DSRPLUS
; 1035        U 1454  3 %THEN
; 1036        U 1455  3
; 1037        U 1456  3             !
; 1038        U 1457  3             ! /OPTIONS - input file is an options file
; 1039        U 1458  3             !
; 1040        U 1459  3             IF  CLI$PRESENT (%ASCID'OPTIONS')
; 1041        U 1460  3             THEN
; 1042        U 1461  3                 BEGIN
; 1043        U 1462  3                 !
; 1044        U 1463  3                 ! Make sure /BOOK_IDENTIFIER was not also specified.
; 1045        U 1464  3                 ! Make sure options file is last in concatenated list.
; 1046        U 1465  3                 !
; 1047        U 1466  3
; 1048        U 1467  3                 IF  CLI$PRESENT (%ASCID'BOOK_IDENTIFIER')
; 1049        U 1468  3                 THEN
; 1050        U 1469  3                     SIGNAL (INDEX$_IGNORED, 1, %ASCID'BOOK_IDENTIFIER', INDEX$_CONFQUAL);
; 1051        U 1470  3
; 1052        U 1471  3                 IF  STATUS EQL CLI$_CONCAT
; 1053        U 1472  3                 THEN
; 1054        U 1473  3                     BEGIN
; 1055        U 1474  3                     !
; 1056        U 1475  3                     ! Current input file concatenated to next - error.
; 1057        U 1476  3                     !
; 1058        U 1477  3                     CLI$GET_VALUE (%ASCID'INPUT', VALUE_STR);
; 1059        U 1478  3                     SIGNAL (INDEX$_IGNORED, 1, VALUE_STR, INDEX$_NOLIST);
; 1060        U 1479  3                     END;
; 1061        U 1480  3
; 1062        U 1481  3                 !
; 1063        U 1482  3                 ! Process options file and exit loop
; 1064        U 1483  3                     !
; 1065        U 1484  3                     OPTIONS_FILE ();
; 1066        U 1485  3                     EXITLOOP;
; 1067        U 1486  3                     END;
; 1068        U 1487  3
; 1069        U 1488  3                 !
; 1070        U 1489  3                 ! Process /BOOK_IDENTIFIER qualifier if present
; 1071        U 1490  3                 !
; 1072        U 1491  3
; 1073        U 1492  3                 PARSE_BOOK ();
; 1074        U 1493  3
; 1075          1494  3   %FI
; 1076          1495  3
; 1077          1496  3                 !
; 1078          1497  3                 ! Process this input file.
; 1079          1498  3                 !
; 1080          1499  3                 ! NDXINP () will call MAKNDX to generate an output index
; 1081          1500  3                 ! if this input file is not concatenated to the previous one.
; 1082          1501  3                 !
; 1083          1502  3                 NDXINP ();
; 1084          1503  3
; 1085          1504  3                 CMDBLK [NDX$V_INPUT_CONCAT] = (.STATUS EQL CLI$_CONCAT);
; 1086          1505  2                 END;
; 1087          1506  2
; 1088          1507  2             !
; 1089          1508  2             ! Generate last output index and clean up
; 1090          1509  2             !
; 1091          1510  2             MAKNDX ();
; 1092          1511  2
; 1093          1512  2             RETURN (.TERMINATION_STATUS OR STS$M_INHIB_MSG);
; 1094          1513  1             END;


                                                             .TITLE  NDXVMS NDXVMS -- DSRINDEX/INDEX Command line in
;                                                                            terface
                                                             .IDENT  \V04-000\

                                                             .PSECT  _LIB$KEY1$,NOWRT,  SHR,  PIC,1

                                             00000 ;TPA$KEYST0
                                                   U.9:      .BLKB   0
                    47  4E  49  4E  4E  55  52  00000 ;TPA$KEYST
                                                   U.11:     .ASCII  \RUNNING\                                  ;
                                          FF  00007          .BYTE   -1                                         ;
                                              00008 ;TPA$KEYST0
                                                   U.16:     .BLKB   0
            47  4E  49  4E  4E  55  52  4F  4E  00008 ;TPA$KEYST
                                                   U.18:     .ASCII  \NORUNNING\                                ;
                                          FF  00011          .BYTE   -1                                         ;
                                          FF  00012 ;TPA$KEYFILL
                                                   U.23:     .BYTE   -1                                         ;

                                                             .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                                             00000 NUMBER_STATE::
                                                             .BLKB   0
                                      41F3  00000 ;TPA$TYPE
```

```
                                          U.2:     .WORD    16883                              ;
                        00000000* 00002  ;TPA$ADDR
                                          U.3:     .LONG    <<QUALIFIER_VALUE-U.3>-4>          ;
                             15F7  00006 ;TPA$TYPE
                                          U.4:     .WORD    5623                               ;
                             FFFF  00008 ;TPA$TARGET
                                          U.5:     .WORD    -1                                 ;
                             15F7  0000A ;TPA$TYPE
                                          U.6:     .WORD    5623                               ;
                             FFFF  0000C ;TPA$TARGET
                                          U.7:     .WORD    -1                                 ;
                                   0000E           .BLKB    2
                                   00010 PAGE_STATE::
                                                   .BLKB    0
                             8300  00010 ;TPA$TYPE
                                          U.12:    .WORD    -32000                             ;
                               01  00012 ;TPA$FLAGS2
                                          U.13:    .BYTE    1                                  ;
                         00000000  00013 ;TPA$PARAM
                                          U.14:    .LONG    0                                  ;
                        00000000V 00017 ;TPA$ACTION
                                          U.15:    .LONG    <<ENTER_PAGE-U.15>-4>              ;
                             8701  0001B ;TPA$TYPE
                                          U.19:    .WORD    -30975                             ;
                               01  0001D ;TPA$FLAGS2
                                          U.20:    .BYTE    1                                  ;
                         00000001  0001E ;TPA$PARAM
                                          U.21:    .LONG    1                                  ;
                        00000000V 00022 ;TPA$ACTION
                                          U.22:    .LONG    <<ENTER_PAGE-U.22>-4>              ;
                             15F7  00026 ;TPA$TYPE
                                          U.24:    .WORD    5623                               ;
                             FFFF  00028 ;TPA$TARGET
                                          U.25:    .WORD    -1                                 ;

                                                   .PSECT   _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                                   00000 NUMBER_KEY::
                                                   .BLKB    0
                                   00000 ;TPA$KEY0
                                          U.1:     .BLKB    0
                                   00000 PAGE_KEY::
                                                   .BLKB    0
                                   00000 ;TPA$KEY0
                                          U.8:     .BLKB    0
                            0000*  00000 ;TPA$KEY
                                          U.10:    .WORD    <U.9-U.8>                          ;
                            0000*  00002 ;TPA$KEY
                                          U.17:    .WORD    <U.16-U.8>                         ;

                                                   .PSECT   $PLIT$,NOWRT,NOEXE,2

                 00  00  00  45  4E  49  4C  24  00000 P.AAB:  .ASCII   \$LINE\<0><0><0>                ;
                                 010E0005  00008 P.AAA:  .LONG    17694725                        ;
                                 00000000' 0000C         .ADDRESS P.AAB                           ;
00  45  47  41  50  5F  52  45  50  5F  53  45  4E  49  4C  00010 P.AAD:  .ASCII   \LINES_PER_PAGE\<0><0>    ;
                                       00  0001F                                                  ;
```

```
                                                   010E000E  00020 P.AAC:   .LONG    17694734
                                                  00000000'  00024          .ADDRESS P.AAD
00  45  47  41  50  5F  52  45  50  5F  53  45  4E  49  4C   00028 P.AAF:   .ASCII   \LINES_PER_PAGE\<0><0>
                                                         00  00037
                                                   010E000E  00038 P.AAE:   .LONG    17694734
                                                  00000000'  0003C          .ADDRESS P.AAF
00  45  47  41  50  5F  52  45  50  5F  53  45  4E  49  4C   00040 P.AAH:   .ASCII   \LINES_PER_PAGE\<0><0>
                                                         00  0004F
                                                   010E000E  00050 P.AAG:   .LONG    17694734
                                                  00000000'  00054          .ADDRESS P.AAH
                        00  45  56  52  45  53  45  52       00058 P.AAJ:   .ASCII   \RESERVE\<0>
                                                   010E0007  00060 P.AAI:   .LONG    17694727
                                                  00000000'  00064          .ADDRESS P.AAJ
                        00  45  56  52  45  53  45  52       00068 P.AAL:   .ASCII   \RESERVE\<0>
                                                   010E0007  00070 P.AAK:   .LONG    17694727
                                                  00000000'  00074          .ADDRESS P.AAL
00  4E  4F  49  54  41  43  49  46  49  54  4E  45  44  49   00078 P.AAN:   .ASCII   \IDENTIFICATION\<0><0>
                                                         00  00087
                                                   010E000E  00088 P.AAM:   .LONG    17694734
                                                  00000000'  0008C          .ADDRESS P.AAN
                                        00  47  4F  4C       00090 P.AAP:   .ASCII   \LOG\<0>
                                                   010E0003  00094 P.AAO:   .LONG    17694723
                                                  00000000'  00098          .ADDRESS P.AAP
                        00  00  54  55  50  54  55  4F       0009C P.AAR:   .ASCII   \OUTPUT\<0><0>
                                                   010E0006  000A4 P.AAQ:   .LONG    17694726
                                                  00000000'  000A8          .ADDRESS P.AAR
                        00  00  54  55  50  54  55  4F       000AC P.AAT:   .ASCII   \OUTPUT\<0><0>
                                                   010E0006  000B4 P.AAS:   .LONG    17694726
                                                  00000000'  000B8          .ADDRESS P.AAT
            53  52  45  42  4D  55  4E  5F  45  47  41  50   000BC P.AAV:   .ASCII   \PAGE_NUMBERS\
                                                   010E000C  000C8 P.AAU:   .LONG    17694732
                                                  00000000'  000CC          .ADDRESS P.AAV
            53  52  45  42  4D  55  4E  5F  45  47  41  50   000D0 P.AAX:   .ASCII   \PAGE_NUMBERS\
                                                   010E000C  000DC P.AAW:   .LONG    17694732
                                                  00000000'  000E0          .ADDRESS P.AAX
                        00  45  52  49  55  51  45  52       000E4 P.AAZ:   .ASCII   \REQUIRE\<0>
                                                   010E0007  000EC P.AAY:   .LONG    17694727
                                                  00000000'  000F0          .ADDRESS P.AAZ
                        00  45  52  49  55  51  45  52       000F4 P.ABB:   .ASCII   \REQUIRE\<0>
                                                   010E0007  000FC P.ABA:   .LONG    17694727
                                                  00000000'  00100          .ADDRESS P.ABB
                        00  00  00  54  55  50  4E  49       00104 P.ABD:   .ASCII   \INPUT\<0><0><0>
                                                   010E0005  0010C P.ABC:   .LONG    17694725
                                                  00000000'  00110          .ADDRESS P.ABD

                                                                            .PSECT   $OWN$,NOEXE,2

                                                       0000  00000 VALUE_STR:
                                                                            .WORD    0
                                                      02 0E  00002          .BYTE    14, 2
                                                   00000000  00004          .LONG    0
                                                       0000  00008 OPTIONS_STR:
                                                                            .WORD    0
                                                      02 0E  0000A          .BYTE    14, 2
                                                   00000000  0000C          .LONG    0
                                                             00010 QUALIFIER_VALUE:
                                                                            .BLKB    4
```

I 14

```
                 00000001  00014 TERMINATION_STATUS:
                                 .LONG    1                                              ;

                                 .EXTRN   DSRINDEX$_BADLOGIC
                                 .EXTRN   DSRINDEX$_BADVALUE
                                 .EXTRN   DSRINDEX$_INSVIRMEM
                                 .EXTRN   DSRINDEX$_LINELENG
                                 .EXTRN   DSRINDEX$_NOREF
                                 .EXTRN   DSRINDEX$_OPENIN
                                 .EXTRN   DSRINDEX$_OPENOUT
                                 .EXTRN   DSRINDEX$_TOOMANY
                                 .EXTRN   DSRINDEX$_VALERR
                                 .EXTRN   DSRINDEX$_CANTBAL
                                 .EXTRN   DSRINDEX$_CLOSEQUOT
                                 .EXTRN   DSRINDEX$_CONFQUAL
                                 .EXTRN   DSRINDEX$_CTRLCHAR
                                 .EXTRN   DSRINDEX$_DOESNTFIT
                                 .EXTRN   DSRINDEX$_DUPBEGIN
                                 .EXTRN   DSRINDEX$_EMPTYIN
                                 .EXTRN   DSRINDEX$_IGNORED
                                 .EXTRN   DSRINDEX$_INVINPUT
                                 .EXTRN   DSRINDEX$_INVRECORD
                                 .EXTRN   DSRINDEX$_LASTCONT
                                 .EXTRN   DSRINDEX$_NOBEGIN
                                 .EXTRN   DSRINDEX$_NOEND
                                 .EXTRN   DSRINDEX$_NOINDEX
                                 .EXTRN   DSRINDEX$_NOLIST
                                 .EXTRN   DSRINDEX$_OVERSTRK
                                 .EXTRN   DSRINDEX$_SKIPPED
                                 .EXTRN   DSRINDEX$_SYNTAX
                                 .EXTRN   DSRINDEX$_TEXFILE
                                 .EXTRN   DSRINDEX$_TOODEEP
                                 .EXTRN   DSRINDEX$_TOOFEW
                                 .EXTRN   DSRINDEX$_TRUNCATED
                                 .EXTRN   DSRINDEX$_COMPLETE
                                 .EXTRN   DSRINDEX$_CREATED
                                 .EXTRN   DSRINDEX$_IDENT
                                 .EXTRN   DSRINDEX$_PROCFILE
                                 .EXTRN   DSRINDEX$_TEXT, DSRINDEX$_TEXTD
                                 .EXTRN   DSRINDEX$_TMS11
                                 .EXTRN   TAB, TMSCOL, MAXLIN
                                 .EXTRN   CLI$_CONCAT, CLI$_PRESENT
                                 .EXTRN   CLI$_NEGATED, CLI$_DEFAULTED
                                 .EXTRN   CLI$_ABSENT, CMDBLK
                                 .EXTRN   CHRSIZ, CHRSZA, CHRSZE
                                 .EXTRN   NDXVRL, NDXVRP, NDXINI
                                 .EXTRN   NDXINP, MAKNDX, CLI$PRESENT
                                 .EXTRN   CLI$GET_VALUE, LIB$TPARSE

                                 .PSECT   $CODE$,NOWRT,2

                      0FFC 00000 .ENTRY   NDXCLI, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-   ; 0800
                                          R11                                           ;
     5B 00000000'  EF   9E 00002 MOVAB    NUMBER_STATE, R11                             ;
     5A 00000000'  EF   9E 00009 MOVAB    NUMBER_KEY, R10                               ;
     59 00000000G  8F   D0 00010 MOVL     #DSRINDEX$_BADVALUE, R9                       ;
     58 00000000G  00   9E 00017 MOVAB    LIB$STOP, R8                                  ;
```

```
                     57 00000000G   00   9E 0001E        MOVAB   CLI$PRESENT, R7                          ;
                     56 00000000G   00   9E 00025        MOVAB   CLI$GET_VALUE, R6                        ;
                     55 00000000'   EF   9E 0002C        MOVAB   P.AAA, R5                                ;
                     54 00000000'   EF   9E 00033        MOVAB   VALUE_STR, R4                            ;
                     53 00000000G   EF   9E 0003A        MOVAB   CMDBLK, R3                               ;
                     6D      0272   CF   DE 00041        MOVAL   30$, (FP)                                ; 0832
          00000000G  EF             00   FB 00046        CALLS   #0, NDXINI                               ; 0840
                               48   A3   9F 0004D        PUSHAB  CMDBLK+72                                ; 0845
                                    55   DD 00050        PUSHL   R5                                       ;
                     66             02   FB 00052        CALLS   #2, CLI$GET_VALUE                        ;
               01    A3             04   8A 00055        BICB2   #4, CMDBLK+T                             ; 0857
          00000000G  EF 00000000G   EF   9E 00059        MOVAB   CHRSZA, CHRSIZ                           ; 0878
               0C    A3             22   D0 00064        MOVL    #34, CMDBLK+12                           ; 0908
               C4    A3 00010001    8F   D0 00068        MOVL    #65537, CMDBLK+4                         ; 0877
               1C    A3        0C   A3   D0 00070        MOVL    CMDBLK+12, CMDBLK+28                     ; 0956
               01    A3             10   8A 00075        BICB2   #16, CMDBLK+1                            ; 1023
               01              04   A3   B1 00079        CMPW    CMDBLK+4, #1                             ; 1056
                     11             12      0007D        BNEQ    2$                                       ;
     06        01    A3             04   E1 0007F        BBC     #4, CMDBLK+1, 1$                         ; 1060
               14    A3             34   D0 00084        MOVL    #52, CMDBLK+20                           ; 1062
                     0A             11      00088        BRB     3$                                       ;
               14    A3             37   D0 0008A 1$:    MOVL    #55, CMDBLK+20                           ; 1064
                     04             11      0008E        BRB     3$                                       ; 1056
               14    A3             36   D0 00090 2$:    MOVL    #54, CMDBLK+20                           ; 1068
                               18   A5   9F 00094 3$:    PUSHAB  P.AAC                                    ; 1070
                     67             01   FB 00097        CALLS   #1, CLI$PRESENT                          ;
                     7B             50   E9 0009A        BLBC    R0, 9$                                   ;
                     04        06   A3   B1 0009D        CMPW    CMDBLK+6, #4                             ; 1077
                     1A             12      000A1        BNEQ    4$                                       ;
                        00000000G   8F   DD 000A3        PUSHL   #DSRINDEX$_CONFQUAL                      ; 1082
                               30   A5   9F 000A9        PUSHAB  P.AAE                                    ;
                                    01   DD 000AC        PUSHL   #1                                       ;
                        00000000G   8F   DD 000AE        PUSHL   #DSRINDEX$_IGNORED                       ;
          00000000G  00             04   FB 000B4        CALLS   #4, LIB$SIGNAL                           ;
                     5B             11      000BB        BRB     9$                                       ;
                               10   A4   D4 000BD 4$:    CLRL    QUALIFIER_VALUE                          ; 1085
                                    54   DD 000C0        PUSHL   R4                                       ; 1086
                               48   A5   9F 000C2        PUSHAB  P.AAG                                    ;
                     66             02   FB 000C5        CALLS   #2, CLI$GET_VALUE                        ;
                                    5A   DD 000C8        PUSHL   R10                                      ; 1088
                             0810   8F   BB 000CA        PUSHR   #^M<R4,R11>                              ;
          00000000V  EF             03   FB 000CE        CALLS   #3, CALL_TPARSE                          ;
                     09             50   E8 000D5        BLBS    R0, 5$                                   ;
                                    54   DD 000D8        PUSHL   R4                                       ; 1090
                                    01   DD 000DA        PUSHL   #1                                       ;
                                    59   DD 000DC        PUSHL   R9                                       ;
                     68             03   FB 000DE        CALLS   #3, LIB$STOP                             ;
               14    A3        10   A4   D0 000E1 5$:    MOVL    QUALIFIER_VALUE, CMDBLK+20               ; 1092
                     50        14   A3   D0 000E6        MOVL    CMDBLK+20, R0                            ; 1095
                     0F             50   D1 000EA        CMPL    R0, #15                                  ;
                     06             18      000ED        BGEQ    6$                                       ;
                     01        04   A3   B1 000EF        CMPW    CMDBLK+4, #1                             ; 1096
                     14             13      000F3        BEQL    8$                                       ;
                     19             50   D1 000F5 6$:    CMPL    R0, #25                                  ; 1099
                     06             18      000F8        BGEQ    7$                                       ;
                     01        04   A3   B1 000FA        CMPW    CMDBLK+4, #1                             ; 1100
                     09             12      000FE        BNEQ    8$                                       ;
```

```
                 00000000G  8F              50  D1 00100 7$:    CMPL    R0, #MAXLIN                              ; 1102
                                            0F  15 00107        BLEQ    9$                                       ;
                            00000000G  8F   DD  00109 8$:       PUSHL   #DSRINDEX$_VALERR                        ; 1104
                                            54  DD 0010F        PUSHL   R4                                       ;
                                            01  DD 00111        PUSHL   #1                                       ;
                                            59  DD 00113        PUSHL   R9                                       ;
                            68              04  FB 00115        CALLS   #4, LIB$STOP                             ;
                       10   A3              02  D0 00118 9$:    MOVL    #2, CMDBLK+16                            ; 1119
                            51       06     A3  32 0011C        CVTWL   CMDBLK+6, R1                             ; 1173
                            01              51  B1 00120        CMPW    R1, #1                                   ; 1176
                                            0B  12 00123        BNEQ    10$                                      ;
                            50       0C     A3  D0 00125        MOVL    CMDBLK+12, R0                            ; 1178
                            50       10 B340    3E 00129        MOVAW   @CMDBLK+16[R0], R0                       ;
                                            0F  11 0012E        BRB     11$                                      ; 1179
                            03              51  B1 00130 10$:   CMPW    R1, #3                                   ; 1183
                                            13  12 00133        BNEQ    12$                                      ;
              50       0C   A3       10     A3  C1 00135        ADDL3   CMDBLK+16, CMDBLK+12, R0                 ; 1186
                            50       1C     A3  C0 0013B        ADDL2   CMDBLK+28, R0                            ; 1187
                 00000078   8F              50  D1 0013F 11$:   CMPL    R0, #120                                 ;
                                            08  11 00146        BRB     13$                                      ;
                 00000078   8F       0C     A3  D1 00148 12$:   CMPL    CMDBLK+12, #120                          ; 1193
                                            09  15 00150 13$:   BLEQ    14$                                      ;
                            00000000G  8F   DD  00152           PUSHL   #DSRINDEX$_LINELENG                      ; 1195
                            68              01  FB 00158        CALLS   #1, LIB$STOP                             ;
                            63       40     8F  8A 0015B 14$:   BICB2   #64, CMDBLK                              ; 1208
                                     18     A3  D4 0015F        CLRL    CMDBLK+24                                ; 1241
                                     58     A5  9F 00162        PUSHAB  P.AAI                                    ; 1243
                            67              01  FB 00165        CALLS   #1, CLI$PRESENT                          ;
                            3F              50  E9 00168        BLBC    R0, 17$                                  ;
                                     10     A4  D4 0016B        CLRL    QUALIFIER_VALUE                          ; 1246
                                            54  DD 0016E        PUSHL   R4                                       ; 1247
                                     68     A5  9F 00170        PUSHAB  P.AAK                                    ;
                            66              02  FB 00173        CALLS   #2, CLI$GET_VALUE                        ;
                                            5A  DD 00176        PUSHL   R10                                      ; 1249
                                   0810     8F  BB 00178        PUSHR   #^M<R4,R11>                              ;
                 00000000V  EF              03  FB 0017C        CALLS   #3, CALL_TPARSE                          ;
                            09              50  E8 00183        BLBS    R0, 15$                                  ;
                                            54  DD 00186        PUSHL   R4                                       ; 1251
                                            01  DD 00188        PUSHL   #1                                       ;
                                            59  DD 0018A        PUSHL   R9                                       ;
                            68              03  FB 0018C        CALLS   #3, LIB$STOP                             ;
                       14   A3       10     A4  D1 0018F 15$:   CMPL    QUALIFIER_VALUE, CMDBLK+20               ; 1253
                                            0F  15 00194        BLEQ    16$                                      ;
                            00000000G  8F   DD  00196           PUSHL   #DSRINDEX$_VALERR                        ; 1255
                                            54  DD 0019C        PUSHL   R4                                       ;
                                            01  DD 0019E        PUSHL   #1                                       ;
                                            59  DD 001A0        PUSHL   R9                                       ;
                            68              04  FB 001A2        CALLS   #4, LIB$STOP                             ;
                       18   A3       10     A4  D0 001A5 16$:   MOVL    QUALIFIER_VALUE, CMDBLK+24               ; 1257
                       0A   A3       63     8F  9B 001AA 17$:   MOVZBW  #99, CMDB[K+10                           ; 1269
                            63       80     8F  8A 001AF        BICB2   #128, CMDBLK                             ; 1303
                                   0080     C5  9F 001B3        PUSHAB  P.AAM                                    ; 1318
                            67              01  FB 001B7        CALLS   #1, CLI$PRESENT                          ;
                            1B              50  E9 001BA        BLBC    R0, 18$                                  ;
                            00000000G  EF   DD  001BD           PUSHL   NDXVRP                                   ; 1320
                            00000000G  EF   DD  001C3           PUSHL   NDXVRL                                   ;
                                            02  DD 001C9        PUSHL   #2                                       ;
                            00000000G   8F   DD 001CB         PUSHL    #DSRINDEX$_IDENT                          ;
              00000000G  00              04   FB 001D1         CALLS    #4, LIB$SIGNAL                            ;
                                   008C  C5   9F 001D8 18$:    PUSHAB   P.AAO                                     ; 1326
                         67              01   FB 001DC         CALLS    #1, CLI$PRESENT                           ;
                         06              50   E9 001DF         BLBC     R0, 19$                                   ;
                    01   A3              02   88 001E2         BISB2    #2, CMDBLK+1                              ; 1328
                                         04   11 001E6         BRB      20$                                       ;
                    01   A3              02   8A 001E8 19$:    BICB2    #2, CMDBLK+1                              ; 1330
                                   009C  C5   9F 001EC 20$:    PUSHAB   P.AAQ                                     ; 1337
                         67              01   FB 001F0         CALLS    #1, CLI$PRESENT                           ;
                         0F              50   E9 001F3         BLBC     R0, 21$                                   ;
                         63              02   88 001F6         BISB2    #2, CMDBLK                                ; 1340
                                     30  A3   9F 001F9         PUSHAB   CMDBLK+48                                 ; 1342
                                   00AC  C5   9F 001FC         PUSHAB   P.AAS                                     ;
                         66              02   FB 00200         CALLS    #2, CLI$GET_VALUE                         ;
                                         03   11 00203         BRB      22$                                       ; 1337
                         63              02   8A 00205 21$:    BICB2    #2, CMDBLK                                ; 1345
                         63              10   8A 00208 22$:    BICB2    #16, CMDBLK                               ; 1350
                         63        0828  8F   A8 0020B         BISW2    #2088, CMDBLK                             ; 1373
                                   00C0  C5   9F 00210         PUSHAB   P.AAU                                     ; 1381
                         67              01   FB 00214         CALLS    #1, CLI$PRESENT                           ;
              00000000G  8F              50   D1 00217         CMPL     R0, #CLI$_NEGATED                         ; 1384
                                         05   12 0021E         BNEQ     23$                                       ;
                         63              08   8A 00220         BICB2    #8, CMDBLK                                ; 1388
                                         31   11 00223         BRB      25$                                       ;
              00000000G  8F              50   D1 00225 23$:    CMPL     R0, #CLI$_PRESENT                         ; 1390
                                         28   12 0022C         BNEQ     25$                                       ;
                                         54   DD 0022E 24$:    PUSHL    R4                                        ; 1395
                                   00D4  C5   9F 00230         PUSHAB   P.AAW                                     ;
                         66              02   FB 00234         CALLS    #2, CLI$GET_VALUE                         ;
                         1C              50   E9 00237         BLBC     R0, 25$                                   ;
                                         5A   DD 0023A         PUSHL    R10                                       ; 1397
                                     10  AB   9F 0023C         PUSHAB   PAGE_STATE                                ;
                                         54   DD 0023F         PUSHL    R4                                        ;
              00000000V  EF              03   FB 00241         CALLS    #3, CALL_TPARSE                           ;
                         E3              50   E8 00248         BLBS     R0, 24$                                   ;
                                         54   DD 0024B         PUSHL    R4                                        ; 1399
                                         01   DD 0024D         PUSHL    #1                                        ;
                                         59   DD 0024F         PUSHL    R9                                        ;
                         68              03   FB 00251         CALLS    #3, LIB$STOP                              ;
                                         D8   11 00254         BRB      24$                                       ; 1397
                         63              04   8A 00256 25$:    BICB2    #4, CMDBLK                                ; 1416
                                   00E4  C5   9F 00259         PUSHAB   P.AAY                                     ; 1418
                         67              01   FB 0025D         CALLS    #1, CLI$PRESENT                           ;
                         0D              50   E9 00260         BLBC     R0, 26$                                   ;
                         63              04   88 00263         BISB2    #4, CMDBLK                                ; 1421
                                     38  A3   9F 00266         PUSHAB   CMDBLK+56                                 ; 1422
                                   00F4  C5   9F 00269         PUSHAB   P.ABA                                     ;
                         66              02   FB 0026D         CALLS    #2, CLI$GET_VALUE                         ;
                    01   A3              01   88 00270 26$:    BISB2    #1, CMDBLK+1                              ; 1428
                    08   A3              03   B0 00274         MOVW     #3, CMDBLK+8                              ; 1429
                         63              01   8A 00278         BICB2    #1, CMDBLK                                ; 1448
                                     28  A3   9F 0027B 27$:    PUSHAB   CMDBLK+40                                 ; 1450
                                   0104  C5   9F 0027E         PUSHAB   P.ABC                                     ;
                         66              02   FB 00282         CALLS    #2, CLI$GET_VALUE                         ;
                         52              50   D0 00285         MOVL     R0, STATUS                                ;
                         1B              52   E9 00288         BLBC     STATUS, 29$                               ;
                         00000000G  EF          00  FB 0028B          CALLS   #0, NDXINP                                    : 1502
                                                50  D4 00292          CLRL    R0                                            : 1504
                         00000000G  8F          52  D1 00294          CMPL    STATUS, #CLI$_CONCAT                          :
                                                02  12 0029B          BNEQ    28$                                           :
                                                50  D6 0029D          INCL    R0                                            :
        63            01            00          50  F0 0029F 28$:     INSV    R0, #0, #1, CMDBLK                            :
                                                D5  11 002A4          BRB     27$                                           : 1450
                         00000000G  EF          00  FB 002A6 29$:     CALLS   #0, MAKNDX                                    : 1510
                  50          14    A4 10000000 8F  C9 002AD          BISL3   #268435456, TERMINATION_STATUS, R0            : 1512
                                                04 002B6              RET                                                   : 1513
                                              0000 002B7 30$:         .WORD   Save nothing                                  : 0832
                                                7E  D4 002B9          CLRL    -(SP)                                         :
                                                5E  DD 002BB          PUSHL   SP                                            :
                                    7E      04  AC  7D 002BD          MOVQ    4(AP), -(SP)                                  :
                         00000000V  EF          03  FB 002C1          CALLS   #3, CONDITION_HANDLER                         :
                                                04 002C8              RET                                                   :

; Routine Size:  713 bytes,    Routine Base:  $CODE$ + 0000
```

```
%SBTTL 'CONDITION_HANDLER - Main program condition handler - sets termination status'
ROUTINE CONDITION_HANDLER (SIG : REF BLOCK [, BYTE], MCH : REF BLOCK [, BYTE]) =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is enabled by NDXCLI as a condition handler.
!       Whenever a signal is generated, the signal severity is examined.
!       If the condition is more severe than any previous condition,
!       (success, warning, error, severe error) the severity is recorded
!       in termination status which is the condition severity. NDXCLI
!       returns the value of TERMINATION_STATUS as the program status
!       which will set the value of the DCL $STATUS variable.
!
! FORMAL PARAMETERS:
!
!       SIG                     - address of signal array
!       MCH                     - address of mechanism array
!
! IMPLICIT INPUTS:
!
!       TERMINATION_STATUS      - current termination severity
!
! IMPLICIT OUTPUTS:
!
!       TERMINATION_STATUS      - may be set to the severity level in the
!                                 signalled condition if it is more severe
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       SS$_RESIGNAL
!
! SIDE EFFECTS:
!
!       None
!--
    BEGIN

    BIND
        SIGNALLED_CONDITION = SIG [CHF$L_SIG_NAME] : BLOCK [, BYTE];

    SELECTONE .SIGNALLED_CONDITION [STS$V_SEVERITY] OF
        SET

        [STS$K_WARNING]:
            IF .TERMINATION_STATUS EQL STS$K_SUCCESS
            THEN
                !
                ! A warning changes the termination status only if it was
                ! 'success' previously.
                !
                TERMINATION_STATUS = STS$K_WARNING;

        [STS$K_ERROR]:
            IF .TERMINATION_STATUS LSS STS$K_ERROR
            THEN
```

```
; 1153        1571  2                    !
; 1154        1572  2                    ! An error status changes the termination status only if it
; 1155        1573  2                    ! was 'success' or 'warning' previously.
; 1156        1574  2                    !
; 1157        1575  2                    TERMINATION_STATUS = STS$K_ERROR;
; 1158        1576  2
; 1159        1577  2          [STS$K_SEVERE]:                              ! Severe error
; 1160        1578  2              TERMINATION_STATUS = STS$K_SEVERE;       ! set the termination status
; 1161        1579  2
; 1162        1580  2          [OTHERWISE]:                                 ! Success or Informational
; 1163        1581  2              ;                                        ! Do nothing
; 1164        1582  2
; 1165        1583  2          TES;
; 1166        1584  2
; 1167        1585  2      RETURN SS$_RESIGNAL;                             ! Continue processing condition
; 1168        1586  1      END;



                                                      0004 00000 CONDITION_HANDLER:
                                                                      .WORD   Save R2                                       ; 1515
                                      52 00000000'   EF   9E 00002        MOVAB   TERMINATION_STATUS, R2
                     50        04     AC             04   C1 00009        ADDL3   #4, SIG, R0                                   ; 1554
                                      07             60   93 0000E        BITB    (R0), #7                                      ; 1559
                                                     09   12 00011        BNEQ    1$
                                      01             62   D1 00013        CMPL    TERMINATION_STATUS, #1                        ; 1560
                                                     1F   12 00016        BNEQ    3$
                                                     62   D4 00018        CLRL    TERMINATION_STATUS                            ; 1566
                                                     1B   11 0001A        BRB     3$                                            ; 1560
          02         60               03             00   ED 0001C 1$:    CMPZV   #0, #3, (R0), #2                              ; 1568
                                                     0A   12 00021        BNEQ    2$
                                      02             62   D1 00023        CMPL    TERMINATION_STATUS, #2                        ; 1569
                                                     0F   18 00026        BGEQ    3$
                                      62             02   D0 00028        MOVL    #2, TERMINATION_STATUS                        ; 1575
                                                     0A   11 0002B        BRB     3$                                            ; 1569
          04         60               03             00   ED 0002D 2$:    CMPZV   #0, #3, (R0), #4                              ; 1577
                                                     03   12 00032        BNEQ    3$
                                      62             04   D0 00034        MOVL    #4, TERMINATION_STATUS                        ; 1578
                                      50       0918  8F   3C 00037 3$:    MOVZWL  #2328, R0                                     ; 1585
                                                          04 0003C        RET                                                   ; 1586

; Routine Size:  61 bytes,    Routine Base:  $CODE$ + 02C9
```

```
%SBTTL 'CALL_TPARSE -- Invoke TPARSE to process qualifier values'
ROUTINE CALL_TPARSE (STRING : REF $STR_DESCRIPTOR (), STATE_TAB, KEY_TAB) =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine calls TPARSE to parse the given string with
!       the given state and key tables.
!
! FORMAL PARAMETERS:
!
!       STRING    - Address of a string descriptor of string to parse
!       STATE_TAB - Address of TPARSE state tables
!       KEY_TAB   - Address of TPARSE key tables
!
! IMPLICIT INPUTS:
!
!       None.
!
! IMPLICIT OUTPUTS:
!
!       None.
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Returns completion code of LIB$TPARSE
!
! SIDE EFFECTS:
!
!       None.
!--

   BEGIN

   LOCAL
       TPARSE_BLOCK : BLOCK [TPA$K_LENGTH0, BYTE];

   !
   ! Initialize the TPARSE parameter block
   !
   TPARSE_BLOCK [TPA$L_COUNT]     = TPA$K_COUNT0;
   TPARSE_BLOCK [TPA$L_OPTIONS]   = TPA$M_ABBREV;
   TPARSE_BLOCK [TPA$L_STRINGCNT] = .STRING [STR$H_LENGTH];
   TPARSE_BLOCK [TPA$L_STRINGPTR] = .STRING [STR$A_POINTER];
   TPARSE_BLOCK [TPA$L_TOKENCNT]  = 0;
   TPARSE_BLOCK [TPA$L_TOKENPTR]  = 0;
   TPARSE_BLOCK [TPA$L_NUMBER]    = 0;
   TPARSE_BLOCK [TPA$L_PARAM]     = 0;

   !
   ! Parse the string and return parse status
   !
   RETURN LIB$TPARSE (TPARSE_BLOCK, .STATE_TAB, .KEY_TAB);
   END;
```

```
                                               0000 00000 CALL_TPARSE:
                                                                   .WORD    Save nothing                            ; 1588
                            5E          20  C2 00002               SUBL2    #32, SP                                 ;
                                        08  DD 00005               PUSHL    #8                                      ; 1628
                       04   AE          02  D0 00007               MOVL     #2, TPARSE_BLOCK+4                      ; 1629
                            50    04    AC  D0 0000B               MOVL     STRING, R0                              ; 1630
                       08   AE          60  3C 0000F               MOVZWL   (R0), TPARSE_BLOCK+8                    ;
                       0C   AE    04    A0  D0 00013               MOVL     4(R0), TPARSE_BLOCK+12                  ; 1631
                                  10    AE  7C 00018               CLRQ     TPARSE_BLOCK+16                         ; 1632
                                  1C    AE  7C 0001B               CLRQ     TPARSE_BLOCK+28                         ; 1634
                            7E    08    AC  7D 0001E               MOVQ     STATE_TAB, -(SP)                        ; 1640
                                  08    AE  9F 00022               PUSHAB   TPARSE_BLOCK                            ;
                  00000000G 00          03  FB 00025               CALLS    #3, LIB$TPARSE                          ;
                                            04 0002C               RET                                              ; 1641

; Routine Size:  45 bytes,    Routine Base:  $CODE$ + 0306
; 1226          1642  1 %SBTTL 'ENTER_PAGE -- Action routine - enter value for /PAGE_NUMBERS'
; 1227          1643  1 ROUTINE ENTER_PAGE =
; 1228          1644  1 !++
; 1229          1645  1 !
; 1230          1646  1 ! FUNCTIONAL DESCRIPTION:
; 1231          1647  1 !
; 1232          1648  1 !       This routine is called as an action routine by TPARSE.
; 1233          1649  1 !
; 1234          1650  1 !       It save the parameter value passed by TPARSE
; 1235          1651  1 !
; 1236          1652  1 ! FORMAL PARAMETERS:
; 1237          1653  1 !
; 1238          1654  1 !       AP [TPA$L_PARAM] - TRUE if STANDARD page numbers, FALSE otherwise
; 1239          1655  1 !
; 1240          1656  1 ! IMPLICIT INPUTS:
; 1241          1657  1 !
; 1242          1658  1 !       None
; 1243          1659  1 !
; 1244          1660  1 ! IMPLICIT OUTPUTS:
; 1245          1661  1 !
; 1246          1662  1 !       CMDBLK [NDX$V_STANDARD_PAGE] - is set to parameter value
; 1247          1663  1 !
; 1248          1664  1 ! ROUTINE VALUE:
; 1249          1665  1 ! COMPLETION CODES:
; 1250          1666  1 !
; 1251          1667  1 !       TRUE
; 1252          1668  1 !
; 1253          1669  1 ! SIDE EFFECTS:
; 1254          1670  1 !
; 1255          1671  1 !       None
; 1256          1672  1 !
; 1257          1673  1 !--
; 1258          1674  1
; 1259          1675  2     BEGIN
; 1260          1676  2
; 1261          1677  2     BUILTIN
; 1262          1678  2         AP;
; 1263          1679  2
; 1264          1680  2     MAP
; 1265          1681  2         AP : REF BLOCK [, BYTE];
; 1266          1682  2
; 1267          1683  2     CMDBLK [NDX$V_STANDARD_PAGE] = .AP [TPA$L_PARAM];
; 1268          1684  2     RETURN TRUE;
; 1269          1685  1     END;



                                                  0000 00000 ENTER_PAGE:
                                                                   .WORD   Save nothing                              ; 1643
00000000G  EF          01          05      20  AC  F0 00002         INSV    32(AP), #5, #1, CMDBLK                    ; 1683
                                   50          01  D0 0000C         MOVL    #1, R0                                    ; 1684
                                                   04 0000F         RET                                               ; 1685

; Routine Size:  16 bytes,    Routine Base:  $CODE$ + 0333
```

```
1 %SBTTL 'OPEN_ERROR - Handle File Open Errors'
1 GLOBAL ROUTINE OPEN_ERROR (FUNCTION_CODE, PRIMARY_CODE, SECONDARY_CODE, IOB : REF $XPO_IOB ()) =
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is called as an Action Routine to report file open errors
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       FUNCTION_CODE    - XPORT failure action routine function code
1 !       PRIMARY_CODE     - primary failure completion code
1 !       SECONDARY_CODE   - secondary failure completion code
1 !       IOB              - Address of file IOB
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Returns the value of PRIMARY_CODE if success is indicated.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals a fatal error terminating program execution if failure
1 !       is indicated by PRIMARY_CODE.
1 !--
1
2   BEGIN
2
2   BIND
2       FILE_SPEC = .IOB [IOB$A_FILE_SPEC] : $STR_DESCRIPTOR (),
2       RESULTANT =  IOB [IOB$T_RESULTANT] : $STR_DESCRIPTOR ();
2
2   LOCAL
2       FILE_NAME : REF $STR_DESCRIPTOR ();
2
2   !
2   ! Point to best file name
2   !
3   FILE_NAME = (IF   .RESULTANT [STR$H_LENGTH] NEQ 0
3                THEN RESULTANT
2                ELSE FILE_SPEC);
2
2   IF  NOT .PRIMARY_CODE
2   THEN
3       BEGIN
3       !
3       ! File was not opened
3       !
3
```

```
; 1328          1743 3          IF  .IOB [IOB$V_INPUT]
; 1329          1744 3          THEN
; 1330          1745 3              SIGNAL_STOP (INDEX$_OPENIN, 1, .FILE_NAME,
; 1331          1746 3                          .IOB [IOB$G_COMP_CODE], 1, .IOB [IOB$G_2ND_CODE])
; 1332          1747 3          ELSE
; 1333          1748 3              SIGNAL_STOP (INDEX$_OPENOUT, 1, .FILE_NAME,
; 1334          1749 3                          .IOB [IOB$G_COMP_CODE], 1, .IOB [IOB$G_2ND_CODE]);
; 1335          1750 3
; 1336          1751 2          END;
; 1337          1752 2
; 1338          1753 2      RETURN .PRIMARY_CODE;
; 1339          1754 1      END;



                                     0000 00000           .ENTRY  OPEN_ERROR, Save nothing                          ; 1687
                  50        10    AC   D0 00002           MOVL    IOB, R0                                           ; 1723
                            1C    A0   B5 00006           TSTW    28(R0)                                            ; 1732
                                  06   13 00009           BEQL    1$                                                ;
                  51        1C    A0   9E 0000B           MOVAB   28(R0), FILE_NAME                                 ;
                                  04   11 0000F           BRB     2$                                                ;
                  51        04    A0   D0 00011 1$:       MOVL    4(R0), FILE_NAME                                  ;
                  35        08    AC   E8 00015 2$:       BLBS    PRIMARY_CODE, 5$                                  ; 1736
                  16        2E    A0   E9 00019           BLBC    46(R0), 3$                                        ; 1743
                          00DC    C0   DD 0001D           PUSHL   220(R0)                                           ; 1746
                                  01   DD 00021           PUSHL   #1                                                ; 1745
                          00D8    C0   DD 00023           PUSHL   216(R0)                                           ; 1746
                                  51   DD 00027           PUSHL   FILE_NAME                                         ; 1745
                                  01   DD 00029           PUSHL   #1                                                ;
                     00000000G    8F   DD 0002B           PUSHL   #DSRINDEX$_OPENIN                                 ;
                                  14   11 00031           BRB     4$                                                ;
                          00DC    C0   DD 00033 3$:       PUSHL   220(R0)                                           ; 1749
                                  01   DD 00037           PUSHL   #1                                                ; 1748
                          00D8    C0   DD 00039           PUSHL   216(R0)                                           ; 1749
                                  51   DD 0003D           PUSHL   FILE_NAME                                         ; 1748
                                  01   DD 0003F           PUSHL   #1                                                ;
                     00000000G    8F   DD 00041           PUSHL   #DSRINDEX$_OPENOUT                                ;
      00000000G   00                   06   FB 00047 4$:  CALLS   #6, LIB$STOP                                      ;
                  50        08    AC   D0 0004E 5$:       MOVL    PRIMARY_CODE, R0                                  ; 1753
                                       04 00052           RET                                                       ; 1754

; Routine Size:  83 bytes,    Routine Base:  $CODE$ + 0343
```

```
%IF DSRPLUS
%THEN

%SBTTL 'ENTER_MERGE -- Action routine - enter page merging parameter'
ROUTINE ENTER_MERGE =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an action routine by TPARSE.
!
!       It saves the parameter passed by TPARSE.
!
! FORMAL PARAMETERS:
!
!       AP [TPA$L_PARAM] - TRUE if MERGE adjacent pages, FALSE otherwise
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       CMDBLK [NDX$V_PAGE_MERGE] - value is stored here
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       TRUE
!
! SIDE EFFECTS:
!
!       None
!
!--

    BEGIN

    BUILTIN
        AP;

    MAP
        AP : REF BLOCK [, BYTE];

    CMDBLK [NDX$V_PAGE_MERGE] = .AP [TPA$L_PARAM];
    RETURN TRUE;
    END;

%FI
```

```
%IF DSRPLUS
%THEN

%SBTTL 'ENTER_LAYOUT -- Action routine - save value of /LAYOUT qualifier'
ROUTINE ENTER_LAYOUT =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an action routine by TPARSE.
!
!       It stores the parameter passed by TPARSE in the command
!       line information block.
!
! FORMAL PARAMETERS:
!
!       AP [TPA$L_PARAM] - Layout value
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       CMDBLK [NDX$H_LAYOUT] - Value is stored here
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       TRUE
!
! SIDE EFFECTS:
!
!       None
!
!--

    BEGIN

    BUILTIN
        AP;

    MAP
        AP : REF BLOCK [, BYTE];

    CMDBLK [NDX$H_LAYOUT] = .AP [TPA$L_PARAM];
    RETURN TRUE;
    END;

%FI
```

```
%IF DSRPLUS
%THEN

%SBTTL 'ENTER_FORMAT -- Action routine - save value of /FORMAT qualifier'
ROUTINE ENTER_FORMAT =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an action routine by TPARSE.
!
!       It stores the parameter passed by TPARSE in the command line
!       information block.
!
!       If the format is TMS=E, it sets up the correct character size
!       vector in CHRSIZ.
!
!       If the format is TEX=filename, it copies filename to TEX_FILE_NAME
!
! FORMAL PARAMETERS:
!
!       AP [TPA$L_PARAM]        - Format type value
!       AP [TPA$L_STRINGCNT]    - Length of filename for TEX=filename
!       AP [TPA$L_STRINGPTR]    - Pointer to filename for TEX=filename
!
! IMPLICIT INPUTS:
!
!       CHRSZE                  - Address of TMS 'E' character size vector
!
! IMPLICIT OUTPUTS:
!
!       CMDBLK [NDX$H_FORMAT]   - Value stored here
!       CHRSIZ                  - Points to TMS 'E' character size vector
!                                 for TMS=E, points to TEX character size
!                                 vector for TEX=filename.
!       TEX_FILE_NAME           - String descriptor of TEX filename.
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       TRUE
!
! SIDE EFFECTS:
!
!       None
!--
    BEGIN

    BUILTIN
        AP;

    MAP
        AP : REF BLOCK [, BYTE];

    CMDBLK [NDX$H_FORMAT] = .AP [TPA$L_PARAM];

    SELECTONE .CMDBLK [NDX$H_FORMAT] OF
```

```
; 1499          U 1911  1           SET
; 1500          U 1912  1
; 1501          U 1913  1           [TMS11_E]:
; 1502          U 1914  1               CHRSIZ = CHRSZE;                    ! TMS 'E' character set
; 1503          U 1915  1
; 1504          U 1916  1           [TEX]:
; 1505          U 1917  1               BEGIN
; 1506          U 1918  1               CHRSIZ = TEX_CHAR_SIZES;            ! TEX character sizes
; 1507          U 1919  1
; 1508          U 1920  1               $STR_COPY (TARGET = TEX_FILE_NAME,
; 1509          U 1921  1                   STRING = (.AP [TPA$L_STRINGCNT], .AP [TPA$L_STRINGPTR]));
; 1510          U 1922  1
; 1511          U 1923  1               END;
; 1512          U 1924  1
; 1513          U 1925  1           [OTHERWISE]:                            ! Do nothing for DSR or TMS=A
; 1514          U 1926  1               ;
; 1515          U 1927  1
; 1516          U 1928  1           TES;
; 1517          U 1929  1
; 1518          U 1930  1       RETURN TRUE;
; 1519          U 1931  1       END;
; 1520          U 1932  1
; 1521            1933  1 %FI
```

```
%IF DSRPLUS
%THEN

%SBTTL 'ENTER_SORT -- Action routine - enter sort type'
ROUTINE ENTER_SORT =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an action routine by TPARSE.
!
!       The parameter passed by TPARSE is stored in the sort type variable.
!
! FORMAL PARAMETERS:
!
!       AP [TPA$L_PARAM] - Sort type value (TRUE or FALSE)
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       CMDBLK [NDX$V_WORD_SORT] - value stored here
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       TRUE
!
! SIDE EFFECTS:
!
!       None
!
!--

    BEGIN

    BUILTIN
        AP;

    MAP
        AP : REF BLOCK [, BYTE];

    CMDBLK [NDX$V_WORD_SORT] = .AP [TPA$L_PARAM];
    RETURN TRUE;
    END;

%FI
```

```
%IF DSRPLUS
%THEN

%SBTTL 'ENTER_ALPHA -- Action routine - enter nonalpha sort value'
ROUTINE ENTER_ALPHA =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an action routine by TPARSE.
!
!       The parameter passed by TPARSE is stored as the nonalpha sort value.
!
! FORMAL PARAMETERS:
!
!       AP [TPA$L_PARAM] - nonalpha sort value
!
! IMPLICIT INPUTS:
!
!       None
!
! IMPLICIT OUTPUTS:
!
!       CMDBLK [NDX$H_NONALPHA] - value is stored here
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       TRUE
!
! SIDE EFFECTS:
!
!       None
!
!--

   BEGIN

   BUILTIN
       AP;

   MAP
       AP : REF BLOCK [, BYTE];

   CMDBLK [NDX$H_NONALPHA] = .AP [TPA$L_PARAM];
   RETURN TRUE;
   END;

%FI
```

```
L  1 %IF DSRPLUS
U  1 %THEN
U  1
U  1 %SBTTL 'OPTIONS_FILE -- Process options file'
U  1 ROUTINE OPTIONS_FILE : NOVALUE =
U  1 !++
U  1 !
U  1 ! FUNCTIONAL DESCRIPTION:
U  1 !
U  1 !       Parse lines of an options file
U  1 !
U  1 ! FORMAL PARAMETERS:
U  1 !
U  1 !       None
U  1 !
U  1 ! IMPLICIT INPUTS:
U  1 !
U  1 !       CMDBLK [NDX$T_INPUT_FILE] - Options file name
U  1 !       NDXOPTION                 - Address of options file parse tables
U  1 !
U  1 ! IMPLICIT OUTPUTS:
U  1 !
U  1 !       CMDBLK [NDX$T_INPUT_FILE] - Input file name
U  1 !
U  1 ! ROUTINE VALUE:
U  1 ! COMPLETION CODES:
U  1 !
U  1 !       None
U  1 !
U  1 ! SIDE EFFECTS:
U  1 !
U  1 !       None
U  1 !
U  1 !--
U  1
U  1     BEGIN
U  1
U  1     LOCAL
U  1         OPTIOB : $XPO_IOB ();
U  1
U  1     $XPO_IOB_INIT (IOB = OPTIOB);
U  1     $XPO_OPEN (IOB = OPTIOB, FILE_SPEC = CMDBLK [NDX$T_INPUT_FILE],
U  1         DEFAULT = '.OPT', FAILURE = OPEN_ERROR);
U  1
U  1     WHILE $XPO_GET (IOB = OPTIOB) EQL XPO$_NORMAL DO
U  1         BEGIN
U  1
U  1         LOCAL
U  1             CH,
U  1             LEN,
U  1             PTR;
U  1
U  1         !
U  1         ! Strip comments from input line
U  1         !
U  1         PTR = CH$FIND_CH (.OPTIOB [IOB$H_STRING], CH$PTR (.OPTIOB [IOB$A_STRING]), %C'!');
U  1
```

```
IF  NOT CH$FAIL (.PTR)
THEN
    !
    ! Remove '!' and everything after it
    !
    LEN = CH$DIFF (.PTR, CH$PTR (.OPTIOB [IOB$A_STRING]))
ELSE
    LEN = .OPTIOB [IOB$H_STRING];

!
! Remove trailing whitespace
!
PTR = CH$PLUS (CH$PTR (.OPTIOB [IOB$A_STRING]), .LEN - 1);

DECR I FROM .LEN - 1 TO 0 DO
    BEGIN
    CH = CH$RCHAR (.PTR);
    PTR = CH$PLUS (.PTR, -1);

    IF  (.CH NEQ %C' ') AND (.CH NEQ TAB)
    THEN
        EXITLOOP;

    LEN = .I;
    END;

IF  .LEN GTR 0
THEN
    BEGIN
    !
    ! We have something to parse
    !
    $STR_COPY (TARGET = OPTIONS_STR,
        STRING = $STR_CONCAT ('OPTIONS ', (.LEN, .OPTIOB [IOB$A_STRING])));

    IF  NOT CLI$DCL_PARSE (OPTIONS_STR, NDXOPTION)
    THEN
        !
        ! Error parsing input line
        !
        SIGNAL_STOP (INDEX$_SYNTAX, 1, OPTIOB [IOB$T_STRING]);

    !
    ! Get input file name
    !
    CLI$GET_VALUE (%ASCID 'INPUT', CMDBLK [NDX$T_INPUT_FILE]);

    IF  CLI$GET_VALUE (%ASCID'INPUT', VALUE_STR)
    THEN
        !
        ! More than one input file specified.
        !
        SIGNAL (INDEX$_IGNORED, 1, VALUE_STR, INDEX$_NOLIST, 0,
                INDEX$_TEXT, 1, OPTIOB [IOB$T_STRING]);

    !
    ! Process /BOOK_IDENTIFIER
```

```
                        !
                        PARSE_BOOK ();

                        !
                        ! Finally, process the input file
                        !
                        NDXINP ();

                        CMDBLK [NDX$V_INPUT_CONCAT] = TRUE;          ! Next file concatenated to this one
                        END;

                    END;

                $XPO_CLOSE (IOB = OPTIOB);
                END;

        %FI
```

```
; 1755      L 2163  1 %IF DSRPLUS
; 1756      U 2164  1 %THEN
; 1757      U 2165  1
; 1758      U 2166  1 %SBTTL 'PARSE_BOOK -- Parse /BOOK_IDENTIFIER qualifier'
; 1759      U 2167  1 ROUTINE PARSE_BOOK : NOVALUE =
; 1760      U 2168  1 !++
; 1761      U 2169  1 !
; 1762      U 2170  1 ! FUNCTIONAL DESCRIPTION:
; 1763      U 2171  1 !
; 1764      U 2172  1 !       This routine is called to process the /BOOK_IDENTIFIER qualifier
; 1765      U 2173  1 !
; 1766      U 2174  1 ! FORMAL PARAMETERS:
; 1767      U 2175  1 !
; 1768      U 2176  1 !       None
; 1769      U 2177  1 !
; 1770      U 2178  1 ! IMPLICIT INPUTS:
; 1771      U 2179  1 !
; 1772      U 2180  1 !       CMDBLK  - Command line information block
; 1773      U 2181  1 !
; 1774      U 2182  1 ! IMPLICIT OUTPUTS:
; 1775      U 2183  1 !
; 1776      U 2184  1 !       CMDBLK [NDX$T_MASTER_BOOK] - Set to book name if doing a master index
; 1777      U 2185  1 !
; 1778      U 2186  1 ! ROUTINE VALUE:
; 1779      U 2187  1 ! COMPLETION CODES:
; 1780      U 2188  1 !
; 1781      U 2189  1 !       None
; 1782      U 2190  1 !
; 1783      U 2191  1 ! SIDE EFFECTS:
; 1784      U 2192  1 !
; 1785      U 2193  1 !       None
; 1786      U 2194  1 !
; 1787      U 2195  1 !--
; 1788      U 2196  1
; 1789      U 2197  1     BEGIN
; 1790      U 2198  1
; 1791      U 2199  1     IF  .CMDBLK [NDX$V_MASTER]
; 1792      U 2200  1     THEN
; 1793      U 2201  1         BEGIN
; 1794      U 2202  1         !
; 1795      U 2203  1         ! Doing a master index
; 1796      U 2204  1         !
; 1797      U 2205  1
; 1798      U 2206  1         IF  CLI$PRESENT (%ASCID'BOOK_IDENTIFIER')
; 1799      U 2207  1         THEN
; 1800      U 2208  1             !
; 1801      U 2209  1             ! User specified a book name
; 1802      U 2210  1             !
; 1803      U 2211  1             CLI$GET_VALUE (%ASCID'BOOK_IDENTIFIER', CMDBLK [NDX$T_MASTER_BOOK])
; 1804      U 2212  1         ELSE
; 1805      U 2213  1             BEGIN
; 1806      U 2214  1             !
; 1807      U 2215  1             ! Doing a master index and no book identifier specified.
; 1808      U 2216  1             ! Use input file name.
; 1809      U 2217  1             !
; 1810      U 2218  1             LOCAL
; 1811      U 2219  1                 PARSE_SPEC_BLOCK : $XPO_SPEC_BLOCK;
```

```
            IF $XPO_PARSE_SPEC (FILE_SPEC = CMDBLK [NDX$T_INPUT_FILE],
                    SPEC_BLOCK = PARSE_SPEC_BLOCK, FAILURE = 0)
            THEN
                !
                ! Filename parse succeeded. Use filename as book name.
                !
                $STR_COPY (STRING = PARSE_SPEC_BLOCK [XPO$T_FILE_NAME],
                       TARGET = CMDBLK [NDX$T_MASTER_BOOK])
            ELSE
                !
                ! Filename parse failed. Use NULL book name.
                !
                $STR_COPY (STRING = '', TARGET = CMDBLK [NDX$T_MASTER_BOOK]);

            END;
        END
    ELSE
        !
        ! Not doing a master index
        !

        IF  CLI$PRESENT (%ASCID'BOOK_IDENTIFIER')
        THEN
            SIGNAL (INDEX$_IGNORED, 1, %ASCID'BOOK_IDENTIFIER', INDEX$_CONFQUAL);

    END;

%FI
```

```
L 1 %IF DSRPLUS
U 1 %THEN
U 1
U 1 %SBTTL 'PROCESS_TEX_FILE - Process TEX character size file'
U 1 ROUTINE PROCESS_TEX_FILE : NOVALUE =
U 1 !++
U 1 !
U 1 ! FUNCTIONAL DESCRIPTION:
U 1 !
U 1 !       This routine is called to process the TEX character size file.
U 1 !
U 1 ! FORMAL PARAMETERS:
U 1 !
U 1 !       None
U 1 !
U 1 ! IMPLICIT INPUTS:
U 1 !
U 1 !       TEX_FILE_NAME    - String descriptor of TEX character size file name
U 1 !
U 1 ! IMPLICIT OUTPUTS:
U 1 !
U 1 !       TEX_FILE_NAME    - Replaced with best file name during file processing.
U 1 !       TEX_CHAR_INDEX   - Initialized to zero
U 1 !       TEX_CHAR_SIZES   - Initialized to zero
U 1 !       TEX_FILE_LINE_NO- Initialized to one
U 1 !
U 1 ! ROUTINE VALUE:
U 1 ! COMPLETION CODES:
U 1 !
U 1 !       None
U 1 !
U 1 ! SIDE EFFECTS:
U 1 !
U 1 !       None
U 1 !--
U 1     BEGIN
U 1     TEX_FILE_LINE_NO = 1;
U 1
U 1     TEX_CHAR_INDEX = 0;
U 1     INCR I FROM 0 TO 255 DO TEX_CHAR_SIZES [.I] =0;
U 1
U 1     !
U 1     ! Set filename and open the file
U 1     !
U 1     TEX_FAB [FAB$B_FNS] = .TEX_FILE_NAME [STR$H_LENGTH];
U 1     TEX_FAB [FAB$L_FNA] = .TEX_FILE_NAME [STR$A_POINTER];
U 1     $OPEN (FAB = TEX_FAB);
U 1     !
U 1     ! Get the best file name
U 1     !
U 1     IF .TEX_NAM [NAM$B_RSL] NEQ 0
U 1     THEN                                                    ! Use resultant name
U 1         $STR_COPY (TARGET = TEX_FILE_NAME,
U 1             STRING = (.TEX_NAM [NAM$B_RSL], .TEX_NAM [NAM$L_RSA]))
U 1     ELSE                                                    ! No resultant name
U 1         BEGIN
U 1
```

```
            IF .TEX_NAM [NAM$B_ESL] NEQ 0
            THEN                                        ! Use expanded name
                $STR_COPY (TARGET = TEX_FILE_NAME,
                    STRING = (.TEX_NAM [NAM$B_ESL], .TEX_NAM [NAM$L_ESA]));

            END;

        IF NOT .TEX_FAB [FAB$L_STS]
        THEN                                            ! File not open
            SIGNAL_STOP (INDEX$_OPENIN, 1, TEX_FILE_NAME,
                .TEX_FAB [FAB$L_STS], .TEX_FAB [FAB$L_STV]);

        IF NOT $CONNECT (RAB = TEX_RAB)                 ! Connect record stream
        THEN                                            !   - error
            SIGNAL_STOP (INDEX$_OPENIN, 1, TEX_FILE_NAME,
                .TEX_RAB [RAB$L_STS], .TEX_RAB [RAB$L_STV]);

        $GET (RAB = TEX_RAB);                           ! Get first line in file
        $STR_DESC_INIT (DESCRIPTOR = TEX_LINE,
            STRING = (.TEX_RAB [RAB$W_RSZ], .TEX_RAB [RAB$L_RBF]));

        IF .TEX_RAB [RAB$L_STS]
        THEN
            BEGIN
            !
            ! Process TEX character size file
            !
            IF RMS$_EOF NEQ CALL_TPARSE (TEX_LINE, TEX_FILE_STATE, TEX_FILE_KEY)
            THEN
                SIGNAL_STOP (INDEX$_TEXFILE, 2, .TEX_FILE_LINE_NO, TEX_FILE_NAME,
                    INDEX$_SYNTAX, T, TEX_LINE);

            END;

        IF .TEX_CHAR_INDEX LSS 128
        THEN                                            ! Not enough values supplied
            SIGNAL (INDEX$_TEXFILE, 2, .TEX_FILE_LINE_NO, TEX_FILE_NAME, INDEX$_TOOFEW);

        $CLOSE (FAB = TEX_FAB);
        END;

%FI
```

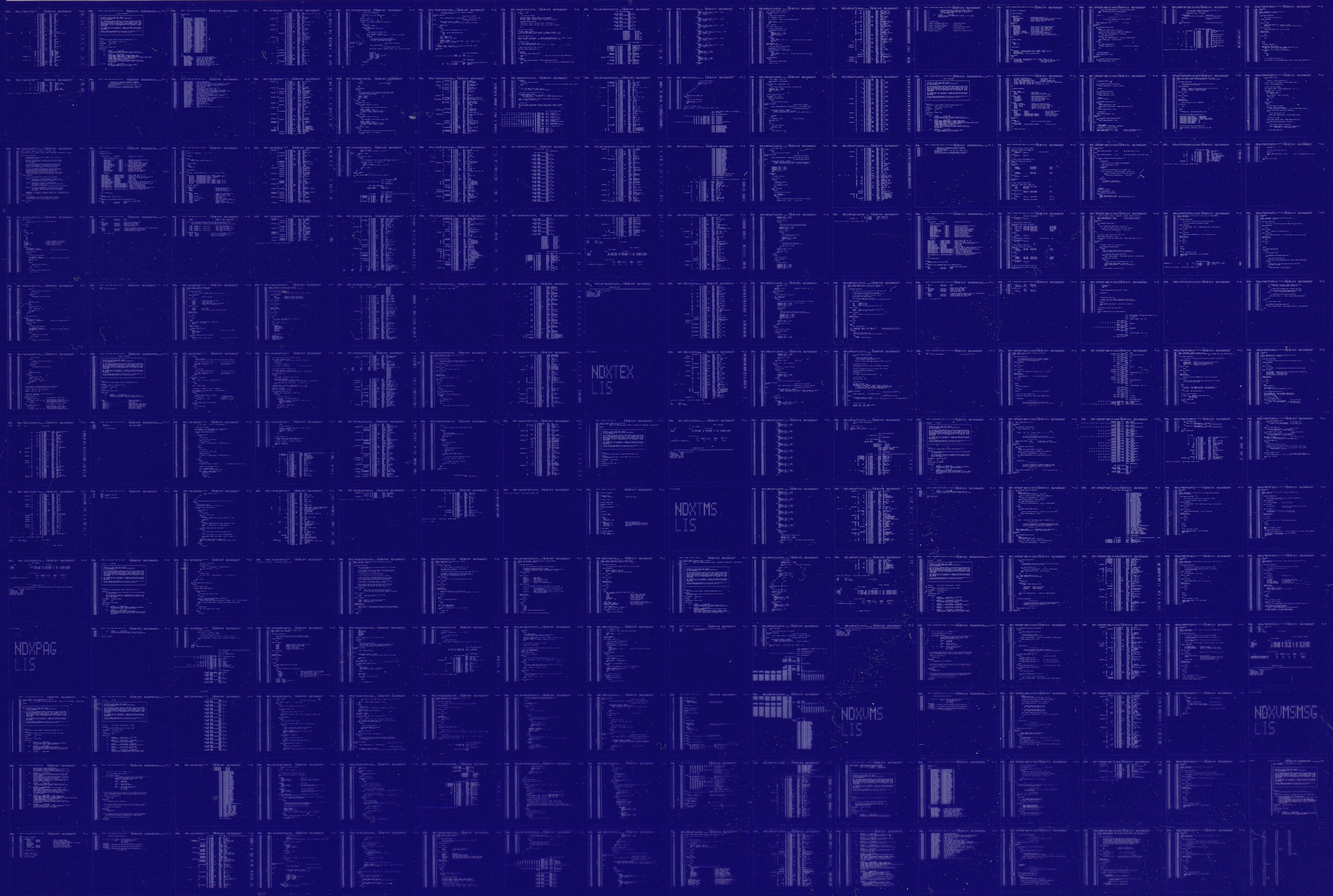
```
L 1 %IF DSRPLUS
U 1 %THEN
U 1
U 1 %SBTTL 'STORE_TEX - Action routine - Store TEX character size'
U 1 ROUTINE STORE_TEX =
U 1 !++
U 1 !
U 1 ! FUNCTIONAL DESCRIPTION:
U 1 !
U 1 !       This routine is called as an action routine by LIB$TPARSE to
U 1 !       store a TEX character size.
U 1 !
U 1 ! FORMAL PARAMETERS:
U 1 !
U 1 !       AP [TPA$L_NUMBER]           - Value to be stored
U 1 !
U 1 ! IMPLICIT INPUTS:
U 1 !
U 1 !       TEX_CHAR_INDEX              - Index into TEX_CHAR_SIZES where next value
U 1 !                                     is to be stored
U 1 !
U 1 ! IMPLICIT OUTPUTS:
U 1 !
U 1 !       TEX_CHAR_SIZES [.TEX_CHAR_INDEX]- Contains value
U 1 !       TEX_CHAR_INDEX                  - Is incremented
U 1 !
U 1 ! ROUTINE VALUE:
U 1 ! COMPLETION CODES:
U 1 !
U 1 !       TRUE
U 1 !
U 1 ! SIDE EFFECTS:
U 1 !
U 1 !       Signals a fatal error if TEX_CHAR_INDEX exceeds 255
U 1 !--
U 1     BEGIN
U 1
U 1     BUILTIN
U 1         AP;
U 1
U 1     MAP
U 1         AP : REF BLOCK [, BYTE];
U 1
U 1     IF .TEX_CHAR_INDEX EGL 256
U 1     THEN
U 1         SIGNAL_STOP (INDEX$_TEXFILE, 2, .TEX_FILE_LINE_NO, TEX_FILE_NAME, INDEX$_TOOMANY);
U 1
U 1     TEX_CHAR_SIZES [.TEX_CHAR_INDEX] = .AP [TPA$L_NUMBER];
U 1     TEX_CHAR_INDEX = .TEX_CHAR_INDEX + 1;
U 1     RETURN TRUE;
U 1     END;
U 1
  1 %FI
```

```
%IF DSRPLUS
%THEN

%SBTTL 'READ_TEX -- Action routine - Read a record from TEX char size file'
ROUTINE READ_TEX =
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an action routine by TPARSE.
!
!       It reads a line from the input file.
!
! FORMAL PARAMETERS:
!
!       None
!
! IMPLICIT INPUTS:
!
!       TEX_RAB                 - RMS RAB to read
!
! IMPLICIT OUTPUTS:
!
!       TEX_IN_BUF              - Contains text of new line
!       TEX_LINE                - Is a string descriptor of new line
!       TEX_FILE_LINE_NO        - Is incremented
!       AP [TPA$L_STRINGCNT]    - Is length of new line
!       AP [TPA$L_STRINGPTR]    - Points to new line
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Returns TRUE if successful
!       Returns RMS$_EOF if end of file encountered
!       Returns FALSE otherwise
!
! SIDE EFFECTS:
!
!       None
!
!--
    BEGIN
    BUILTIN
        AP;

    MAP
        AP : REF BLOCK [, BYTE];

    IF NOT $GET (RAB = TEX_RAB)
    THEN
        RETURN (IF .TEX_RAB [RAB$L_STS] EQL RMS$_EOF THEN RMS$_EOF ELSE FALSE);

    TEX_FILE_LINE_NO = .TEX_FILE_LINE_NO + 1;
    $STR_DESC_INIT (DESCRIPTOR = TEX_LINE,
        STRING = (.TEX_RAB [RAB$W_RSZ], .TEX_RAB [RAB$L_RBF]));

    AP [TPA$L_STRINGCNT] = .TEX_LINE [STR$H_LENGTH];
    AP [TPA$L_STRINGPTR] = .TEX_LINE [STR$A_POINTER];
```

```
; 2053       U 2458  1        RETURN TRUE;
; 2054       U 2459  1        END;
; 2055       U 2460  1
; 2056         2461  1 %FI
; 2057         2462  1
; 2058         2463  1 END                             ! End of module
; 2059         2464  0 ELUDOM



                                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                                     PSECT SUMMARY
;
;       Name                        Bytes                         Attributes
;
;   $OWN$                              24  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _LIB$KEY0$                          4  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   _LIB$STATE$                        42  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   _LIB$KEY1$                         19  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   $PLIT$                            276  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                            918  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                              Library Statistics
;
;                                              -------- Symbols --------          Pages        Processing
;       File                                   Total    Loaded   Percent          Mapped       Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1           9776        20         0            581          00:01.0
;   _$255$DUA28:[SYSLIB]TPAMAC.L32;1              42        25        59             14          00:00.1
;   _$255$DUA28:[SYSLIB]XPORT.L32;1              590       119        20            252          00:00.6



;                                     COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:NDXVMS/OBJ=OBJ$:NDXVMS MSRC$:NDXVMS/UPDATE=(ENH$:NDXVMS)

; Size:           918 code + 365 data bytes
; Run Time:          00:42.1
; Elapsed Time:      01:22.5
; Lines/CPU Min:       3514
; Lexemes/CPU-Min:    45469
; Memory Used:     273 pages
; Compilation Complete
```

0345 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


NDXTEX
LIS

NDXTMS
LIS

NDXPAG
LIS

NDXVMS
LIS

NDXVMSMSG
LIS